

Identifying bias in CNN image classification using image scrambling and
transforms

by

Sai Teja Erukude

B.Tech, Jawaharlal Nehru Technological University Hyderabad, 2020

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Dr. Lior Shamir

Copyright

© Sai Teja Erukude 2024.

Abstract

Convolutional Neural Networks are now prevalent as the primary choice for most machine vision problems due to their superior rate of classification and the availability of user-friendly libraries. These networks effortlessly identify and select features in a non-intuitive data-driven manner, making it difficult to determine which features were most influential in the learning process. That leads to a “black box”, where users cannot know how the image data are analyzed but rely on empirical results. Therefore the decision-making process can be biased by background information that is difficult to detect. Here we discuss examples of such hidden biases and propose techniques for identifying them, methods to distinguish between contextual information and background noise, and explore whether CNNs learn from irrelevant features. One effective approach to identify dataset bias is to classify blank background parts of the images. However, in some situations a blank background in the images is not available, making it more difficult to separate the foreground information from the blank background. Such parts of the image can also be considered contextual learning, not necessarily bias. To overcome this, we propose two approaches that were tested on six different datasets, including natural, synthetic, and hybrid datasets. The first method involves dividing images into smaller, non-overlapping tiles of various sizes, which are then shuffled randomly, disrupting the features of the objects, and making classification more challenging. The second method involves the application of several image transforms, including Fourier, Wavelet transforms, and Median filter, and their combinations. These transforms help recover background noise information used by CNN to classify images. Results indicate that this method can effectively distinguish between contextual information and background noise, and alert on the presence of background noise even without the need to use background information.

Table of Contents

List of Figures	vi
List of Tables	viii
List of Nomenclature	ix
Acknowledgements	x
1 Introduction	1
1.1 Convolutional neural networks (CNNs)	2
1.1.1 Drawbacks of CNNs	3
1.2 Bias in CNN Image Classification	4
1.2.1 Benchmark datasets	4
1.2.2 Medical datasets	5
2 Data and CNN architecture	8
2.1 Natural datasets	9
2.1.1 Imagenette	9
2.1.2 Natural images	10
2.2 Non-natural or synthetic datasets	10
2.2.1 CT scans	11
2.2.2 Coil-20	12
2.2.3 Yale faces	13
2.3 Mixed or Hybrid datasets	14
2.3.1 Caltech 20	14

2.4	CNN architecture	15
3	Identifying bias using background segments and image scrambling	18
3.1	Introduction	18
3.2	Methods	21
3.2.1	Cropping background segments	21
3.2.2	Image scrambling	23
3.3	Results	25
3.3.1	Classification results	25
3.3.2	Cropped background segments results	26
3.3.3	Image scrambling results	28
3.4	Conclusion	30
4	Identifying bias using image transforms	31
4.1	Methods	32
4.1.1	Fourier transform	32
4.1.2	Wavelet transform	35
4.1.3	Median Filter	39
4.1.4	Median filter + Fourier transform	41
4.1.5	Median filter + Wavelet transform	43
4.2	Conclusion	46
4.3	Code repository	47
4.3.1	Files description	47
	Bibliography	49

List of Figures

1.1	A deep neural network representation	2
1.2	A convolutional neural network representation	3
1.3	X-rays images classified correctly by CNNs	6
2.1	Categorization of data	8
2.2	<i>Imagenette</i> dataset with 10 classes	9
2.3	<i>Natural images</i> dataset	10
2.4	CT scans dataset with 4 classes	12
2.5	Coil-20 dataset with 20 classes	13
2.6	Yale faces dataset	14
2.7	Caltech 20 dataset	15
2.8	VGG16 architecture map	16
2.9	VGG16 architecture	17
3.1	20×20 cropped background segment	22
3.2	20×20 cropped yale faces dataset	22
3.3	Images with no background	23
3.4	Tile scrambling technique outputs	24
3.5	Corresponding 20×20 cropped Imagenette dataset	26
3.6	Classification accuracies of cropped segments vs random chance	27
3.7	Corresponding scrambled Imagenette dataset	28
3.8	Classification accuracies of scrambled datasets vs random chance	29
4.1	Fourier transformed image	32

4.2	Fourier transform of 20×20 cropped image	33
4.3	Classification accuracies of Fourier transformed full images	34
4.4	Classification accuracies of Fourier transformed cropped images	34
4.5	Discrete Wavelet transformed image	36
4.6	Classification accuracies of Wavelet transformed full images	37
4.7	Classification accuracies of Wavelet transformed cropped images	37
4.8	Median filtered image	39
4.9	Classification accuracies of Median filtered full images	40
4.10	Classification accuracies of Median filtered cropped images	41
4.11	Median + Fourier transformed image	42
4.12	Classification accuracies of Median and Fourier transformed full images . . .	43
4.13	Classification accuracies of Median and Fourier transformed cropped images	43
4.14	Median + Wavelet transformed image	44
4.15	Classification accuracies of Median and Wavelet transformed full images . . .	45
4.16	Classification accuracies of Median and Wavelet transformed cropped images	45

List of Tables

2.1	Split of Imagenette dataset	9
2.2	Split of <i>Natural images</i> dataset	10
2.3	Split of CT scan dataset	11
2.4	Split of Coil-20 dataset	12
2.5	Split of augmented Yale faces dataset	13
2.6	Split of Caltech 20 dataset	14
3.1	Classification accuracies of raw datasets	25
3.2	Classification accuracies of cropped datasets	26
3.3	Classification accuracies of scrambled datasets	29
4.1	Classification accuracies of Fourier transformed datasets	33
4.2	Classification accuracies of Haar Wavelet transformed datasets	36
4.3	Classification accuracies of Daubechies Wavelet transformed datasets	38
4.4	Classification accuracies of Median filtered datasets	40
4.5	Classification accuracies of Median filtered Fourier transformed datasets	42
4.6	Classification accuracies of Median filtered Wavelet transformed datasets	44

Nomenclature

- CAM - Class Activation Mappings
- CNN - Convolutional Neural Network
- DFT - Discrete Fourier Transform
- DWT - Discrete Wavelet Transform
- FC - Fully connected layer
- FIR - Finite Impulse Response
- ReLU - Rectified Linear Units
- ResNet - Residual Neural Network
- RMSprop - Root Mean Square Propagation
- ROI - Region Of Interest
- VGG16 - Visual Geometry Group 16

Acknowledgments

Above all, I would like to take this opportunity to be eternally grateful to my major advisor, Dr. Lior Shamir, for his outstanding mentorship and belief in my research capabilities. His knowledge and guidance have been indispensable throughout my research, giving me the confidence and shared knowledge that helped me put this thesis together. Dr. Shamir's support and constructive feedback have enriched the quality of my work and eased my path of growth considerably in both personal and professional respect.

This leads me to my committee members, Dr. Mitchell Neilsen and Dr. Torben Amtoft, to whom I give my most sincere thanks, who have so generously supported serving on my committee by devoting considerable time to this research effort.

Lastly, I express my most profound feelings of indebtedness to my family and friends who have shared constant affection and support in this voyage. Their contribution has been fundamental in more ways than one could count. I am deeply grateful to my girlfriend, Jane Mascarenhas, whose continued love, patience, and encouragement have substantially contributed to my persistence and success.

Chapter 1

Introduction

Deep learning could be explained as machine learning methods that rely on deep neural networks with huge numbers of interconnected layers. These have achieved state-of-the-art performance in many other applications ranging from image classification, object detection, speech recognition, and language translation.

Unlike other forms of traditional machine learning methods, deep learning algorithms do not require explicit human involvement in feature extraction, selection, modeling, and testing. Deep learning algorithms can automatically extract and select features from raw data, like images, videos, or text. The term “deep” in deep learning refers to many layers of algorithms, or neural networks. These are very flexible architectures that can learn directly from raw data somewhat like the human brain and, given more data, do so with improved predictive accuracy.

It is also the bedrock technology that can further reach unparalleled feasibility and accuracy in speech-to-text processing, language translation, and object recognition, to name but a few, other applications of artificial intelligence (AI). It is rightly accredited to several recent breakthroughs in the field of AI, including AlphaGo from Google DeepMind, driverless cars, voice assistants, and many more.

Deep learning uses multi-layered artificial neural networks (ANNs), which are networks composed of several “hidden layers” of nodes between the input and output (see Figure 1.1).

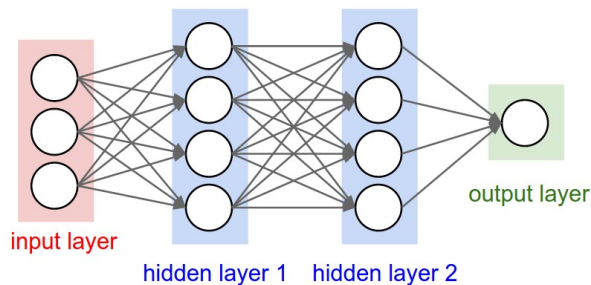


Figure 1.1: *A deep neural network representation¹*

1.1 Convolutional neural networks (CNNs)

Convolutional neural networks, also known as ConvNets have been the most-used architecture in deep learning when it comes to image and video processing. With high intrinsic performance in feature identification and pattern identification, they are suitable for tasks such as object detection, image recognition, pattern identification, and even facial recognition. Some linear algebra practices majorly by matrix multiplication form the basis of how these CNNs search for patterns in images.

Interest in CNNs exploded with the introduction of AlexNet² in 2012. From that day forward, in only three short years, it went from an 8-layer AlexNet to a 152-layer ResNet³. The rapidity with which this occurred speaks to the massive success and popularity that both CNNs and deep learning in general are enjoying.

They work much better for high-dimensional data, including images, speech, or audio inputs, than previous methods relying on the segmentation and manual feature extraction process. They offer an image classification and object recognition approach with a scalable nature and allow for efficient data processing by reducing overfitting risks, though with the potential loss of some information in pooling layers. The main advantage of CNN models is their computational efficiency, which enables them to run on a wide range of devices.

Figure 1.2 represents the major components of CNN: convolutional layers, pooling layers, and fully connected or FC layers. A few of the CNNs are incredibly complex and involve thousands of layers, each layer enhancing the ability of the model. Early layers detect simple features of colors and edges; however, as the layers go higher, the network goes step by step

to detect complex components and finally the whole object. Another strong point is that feature learning is done automatically and without intervention; this was not the case for some of the earlier models. If one trains a CNN on some pictures of cats versus dogs, it automatically learns how to select features that will make dogs distinguishable from cats.

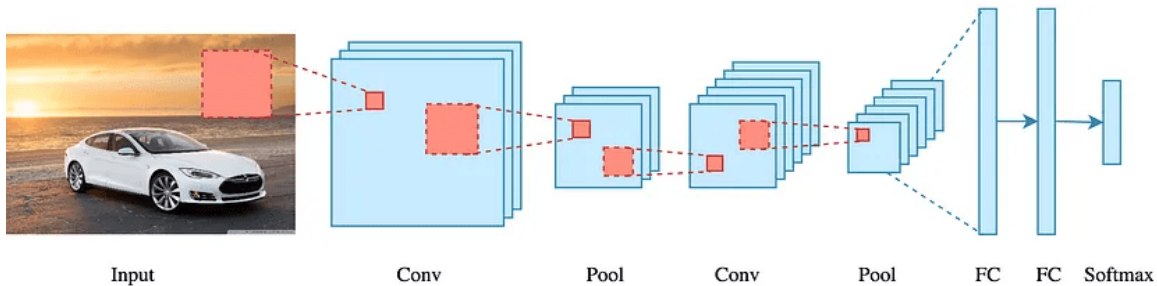


Figure 1.2: *A convolutional neural network representation*⁴

In the hidden layers of a CNN, convolution filters are employed for feature map extraction of input data, which further undergoes processing through pooling layers and finally feeds into the output layer, marking the end of processing by the network. This is a complex process that gives fantastic results to CNNs; at the same time, it is loaded with such a level of complexity that the network seems like a "black box." What goes on within those hidden layers is very obscure, even to the creators, because of the non-intuitive rules at work.

Nowadays, these models stand at the forefront of various kinds of image-related tasks and are further expanded into many other domains such as recommender systems and natural language processing. This is the reason for the versatility and efficiency that keep them at the front line of attention when it comes to wide applicability within deep learning.

1.1.1 Drawbacks of CNNs

With great advantages brought about by CNN, they also bear most of the major limitations. For example, high computation degree: The computation generally involves several GPUs for execution and is a very time-consuming process, not to mention expensive. CNNs apply highly skilled experts who are knowledgeable in multidisciplinary fields to perform comprehensive testing, setting modification, and hyperparameters among many other variables.

Less discussed, however, is probably the most serious drawback: the “black box” nature of CNNs. While CNNs are trained over data samples, updating their weights in a view to do better, the decision-making process remains opaque. The rules that define how classifications are made are convoluted and counterintuitive, and it has proven difficult to interpret the reason behind the decision of a network. The definition of what a CNN really “learns” from the data thus becomes equally difficult to make. In other words, lack of interpretability simply means that CNNs should be used with caution as their classification processes are not certain in CNNs^{5,6}. Section 1.2 will take a deeper dive into this issue.

1.2 Bias in CNN Image Classification

Ideally, a CNN that can achieve high accuracy in classification on benchmark datasets should perform well in the finding of real-world objects. However, several studies proved that many popular datasets were biased and possibly not representative of a model’s performance concerning real-world object recognition.⁷

Neural networks automatically extract and select features from an image in a non-intuitive, data-driven manner. Since the images are unsegmented and may contain irrelevant artifacts, some contributions of these images to the CNN prediction may have nothing to do with the main object of interest. It thus becomes difficult to find those concrete features that have had most of the influence on the network’s learning and decision processes. Therefore, it also can be biased by the background information in such a manner that it is hard to notice.

1.2.1 Benchmark datasets

Classification bias is present in many widely used benchmark datasets. Model and Shamir⁸ conducted an in-depth study comparing dataset biases across various object recognition benchmarks, including COIL-20, COIL-100, NEC Animals, Caltech 101, etc. They employed a method that involved isolating a small, seemingly blank background area from each image

- an area that does not contain any information about the object of interest. This technique was used to detect dataset bias in single-object recognition datasets and to compare the extent of bias across different datasets. The findings revealed that all the datasets tested still achieved classification accuracy above chance levels using these small images, even though the sub-images lacked any visually interpretable information.

The impact of bias is significantly more in object datasets collected in controlled environments, like COIL-20, COIL-100, and NEC Animals, compared to datasets sourced from natural images found online, such as ImageNet⁹. As these datasets achieved classification accuracy well above chance using only seemingly identical background areas indicates the presence of artifacts in the images. This suggests that similar biases may also be present in the foreground object areas, potentially enabling machine vision algorithms to classify images even without properly identifying the objects depicted.

1.2.2 Medical datasets

Majeed et al.¹⁰ encountered similar challenges when they investigated the applicability of CNNs for detecting COVID-19 in chest X-ray images. Their study involved using 12 established CNN architectures in transfer learning mode across three publicly available chest X-ray databases. Additionally, they introduced a custom shallow CNN model, which they trained from scratch. For their experiments, chest X-ray images were input into CNN models without any pre-processing, mirroring how other studies have used these images.

To better understand how these CNNs made their predictions, they conducted a qualitative analysis using a method called Class Activation Mappings (CAM)¹¹. CAMs allow researchers to generate heatmaps of the most discriminating regions and visualize which parts of the input image were most influential in the network’s decision-making process. By mapping the activations from the CNN back to the original image, one can highlight class-specific regions that contributed most significantly to the model’s classification, providing insights into which features were most relevant for detecting COVID-19.

Figure 1.3 illustrates the regions of images that CNN models focused on when making

predictions. It is evident that, in only a few cases, the models concentrated on the frontal chest region (i.e., the lung area), which is where we typically look for signs of COVID-19 and other infections. Instead, these models often focused on areas outside the chest’s frontal view, as seen in the first column of row (b) and the third and fourth columns of row (e) in Figure 5. Additionally, there are noticeable overlaps between the CAM hot spots and textual elements in the images, particularly in the first column of row (b), and the first, third, and fourth columns of row (e).

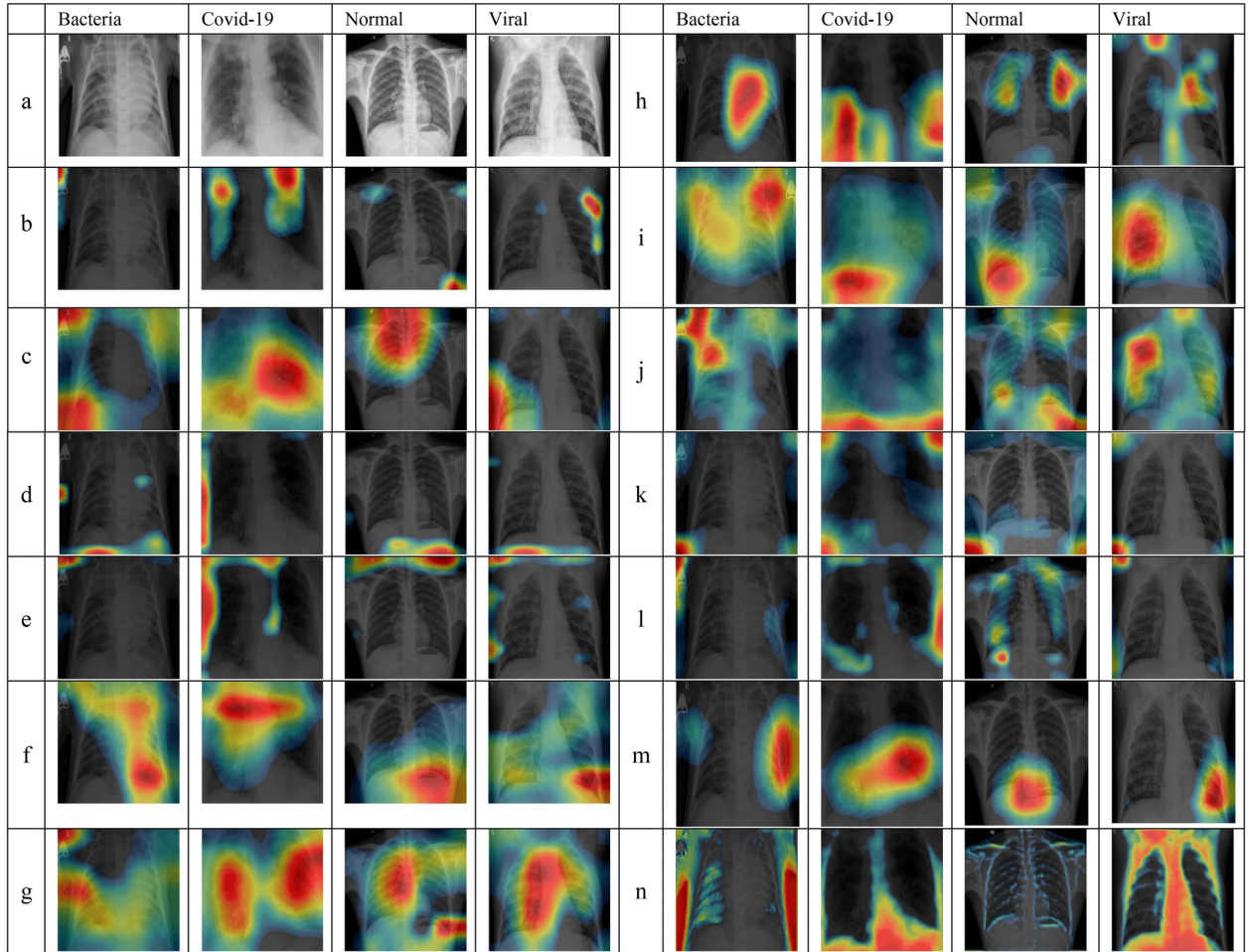


Figure 1.3: *X-rays images classified correctly by CNNs. a Original X-ray, b AlexNet, c GoogleNet, d VGG16, e VGG19, f ResNet18, g ResNet50, h ResNet101, i Inception V3, j InceptionResNet, k DenseNet201, l SqueezeNet, m Xception, and n CNN-X¹⁰*

This is a notable example where almost all of the CNN architectures used areas outside the region of interest (ROI) to make their final classification predictions. This observation leads

us to conclude that directly using X-ray images without pre-processing such as segmenting the ROI and removing background noise, can lead to biased and misleading classification results.

In summary, CNN predictions should be considered cautiously, even if they have high accuracy until the region(s) of the input image used by CNNs that lead to its prediction are visually inspected and approved. It is also demonstrated that CNNs are ‘cheating’ by using artifacts like text or medical device marks and background noise present in the images to build their prediction that has nothing to do with the object of interest.

It is now evident that nearly every dataset contains some level of bias, and this should be taken into account before relying solely on high-accuracy figures. This thesis proposes several techniques for detecting classification bias, even in cases where it is not feasible to isolate segments of the image background.

Chapter 2

Data and CNN architecture

This research was carried out across six distinct datasets to ensure that the findings were valid and robust across different contexts. As shown in Figure 2.1, datasets used in this study can be further categorized into three types: natural datasets 2.1, non-natural or synthetic datasets 2.2 and mixed or hybrid datasets 2.3. Successive sections describe in detail the source of the datasets and the sizes of the training, testing, and validation splits.

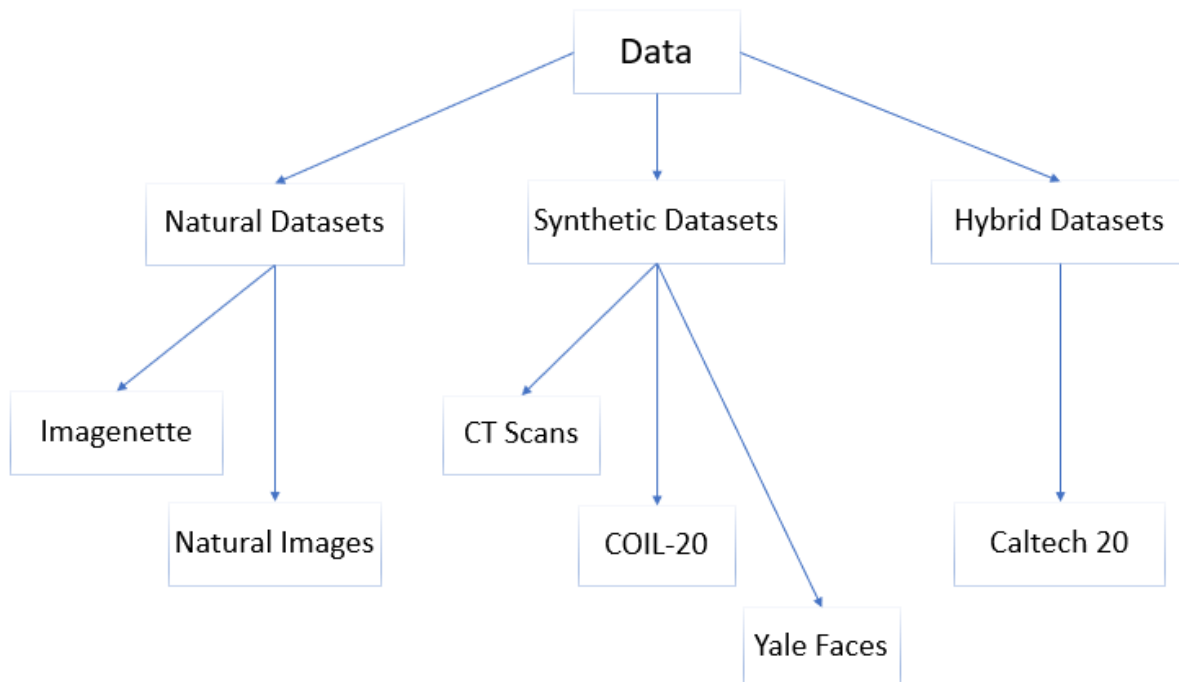


Figure 2.1: *Categorization of data*

2.1 Natural datasets

Natural datasets involve data in which, in general, information is taken from the real world, representing some natural phenomenon or environment. They contain images captured in real life, like photographs of people, animals, objects, and scenes. They have high variability in lighting conditions, angles, backgrounds, and object appearances that help in making models robust for real-world conditions. These often contain a variety of patterns and complex noise and variations that make the training of the model more difficult, but create better applicability for real-world tasks. For example, two such naturally collected datasets used in this work are Imagenette [2.1.1](#) and Natural images [2.1.2](#).

2.1.1 Imagenette

*Imagenette*¹² is a smaller subset of the *ImageNet*⁹ dataset, and it contains 10 different classes. It is great to quickly experiment and fine-tune techniques before applying them to the full-scale *ImageNet* dataset. Table [2.1](#) illustrates the train, test, and val split and the figure [2.2](#) depicts the 10 different classes present in the dataset.

Train	Test	Validation	Total Images
9,469	1,904	2,021	13,394

Table 2.1: *Split of Imagenette dataset*



Figure 2.2: *Imagenette dataset with 10 classes*

2.1.2 Natural images

The *Natural images* dataset consists of images coming from 8 different classes collected from several sources¹³. The classes include airplane, car, cat, dog, flower, fruit, motorbike and person. The class “fruit” will be discarded for the purposes of this research because the background was changed to white, which could introduce biases in the classification process.

Table 2.2 shows the distribution of the dataset across training, testing and validation sets while Figure 2.3 shows all 7 subjects.

Train	Test	Validation	Total Images
3,367	709	711	4,787

Table 2.2: *Split of Natural images dataset*



Figure 2.3: *Natural images dataset*

2.2 Non-natural or synthetic datasets

Non-natural datasets, on the other hand, consist of data that are either synthetically generated or created under controlled conditions. Since these datasets are typically generated in controlled environments or through algorithms, they exhibit less variability compared to

natural datasets. They are frequently used to streamline model training or to emphasize particular characteristics of the data. Designed to address specific tasks or problems, such as recognizing patterns in simplified scenarios, these datasets tend to have simpler features and reduced variability, making them useful for targeted experiments or testing specific hypotheses.

Examples of synthetic datasets include images generated through computer graphics or simulation tools, such as those from 3D rendering engines, virtual environments, or simulations of medical imaging and robotic vision. In this research, we used three such non-natural datasets, which will be described in the following sections. Studies have demonstrated that all three datasets contain some form of bias or background noise, which impacts the learning process of CNNs.

2.2.1 CT scans

This dataset consists of CT scan images of lungs that are categorized into four classes: COVID-19, normal, bacterial pneumonia, and viral pneumonia. Note that these images have many types of artifacts including text, medical device traces, and background noise which will significantly influence the CNN model’s learning and decision-making process¹⁰. Additionally, it was demonstrated that CNNs were using regions/features in the input image that are outside the region of interest (ROI) and have no relation with COVID-19.

Table 2.3 shows the distribution of these images across training, testing, and validation sets, while Figure 2.4 displays examples of the four distinct classes within the dataset.

Train	Test	Validation	Total Images
1,188	224	226	1,638

Table 2.3: *Split of CT scan dataset*

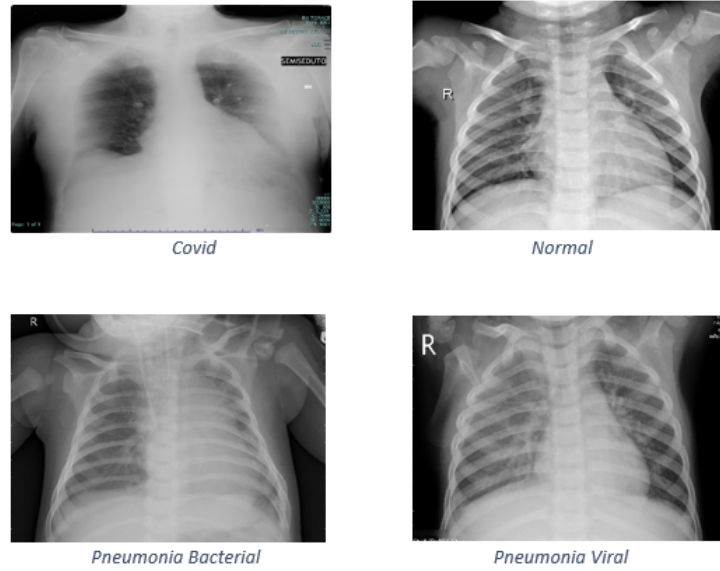


Figure 2.4: *CT scans dataset with 4 classes*

2.2.2 Coil-20

Coil-20, short for Columbia object image library¹⁴ is a repository of gray-scale pictures that contains such twenty objects. All the objects could be precisely put on a turntable. The turntable was turned at 360 degrees until a change of object was appropriately visible from a static camera. Images of the objects were captured in 5-degree angular orientations. Model and Shamir⁸ demonstrated in their study that *COIL-20*, being generated in a controlled environment, contains biases that significantly affect the model’s classification accuracy. They achieved exceptionally high classification accuracy even when the object was absent from the testing image, which means the model has been memorizing patterns from bias or background noise.

Table 2.4 shows the distribution of the data across training, testing, and validation sets, while Figure 2.4 displays examples of the twenty different classes within the dataset.

Train	Test	Validation	Total Images
1,160	140	140	1,440

Table 2.4: *Split of Coil-20 dataset*



Figure 2.5: *Coil-20 dataset with 20 classes*

2.2.3 Yale faces

This dataset is originally from the *Yale Face Database*.^{15;16} There are 165 GIFs of 15 subjects, containing the following facial expressions: happy, sad, with and without glasses, sleepy, normal, wink, surprised, and lighting from different directions. Dhar and Shamir¹⁷ assessed benchmark datasets, including *Yale Faces*, to evaluate CNN performance. Their findings reveal a consistent bias within the dataset, with the CNN achieving notably higher classification accuracy when only a part of the image was used, highlighting the significant extent of this bias.

Figure 2.6 displays all the images of a subject. The GIFs were then converted to JPEG format and augmented with horizontal and vertical flips. Table 2.5 provides the distribution of the augmented dataset across training, testing, and validation sets.

Train	Test	Validation	Total Images
315	90	90	495

Table 2.5: *Split of augmented Yale faces dataset*



Figure 2.6: *Yale faces dataset*

2.3 Mixed or Hybrid datasets

Mixed or hybrid datasets are collections of images that combine both real-world (natural) images and synthetically generated images. Such datasets are designed for leveraging both data types to improve the performance and generalization of models in tasks such as image classification. Hybrid datasets offer a wider variety of scenarios and conditions, which helps models to generalize more effectively. Additionally, the ability to generate synthetic images in large quantities helps mitigate the problem of having limited real-world data.

2.3.1 Caltech 20

The *Caltech 20* database is a subset of the larger *Caltech 256* dataset, which was sourced from Caltech Data and comprises 25,607 images across 256 object categories. For the purpose of this study, only the first 20 object categories were used to simplify and expedite the process. Interestingly, upon closer inspection, we found that the first 20 classes feature a combination of real-world images and either graphically generated or have had their backgrounds altered to a plain white/black backdrop. Therefore, *Caltech 20* is categorized as a mixed dataset throughout this study.

Table 2.6 provides the distribution of the augmented dataset across training, testing, and validation sets, and Figure 2.7 displays all 20 subjects.

Train	Test	Validation	Total Images
1,812	368	360	2,540

Table 2.6: *Split of Caltech 20 dataset*

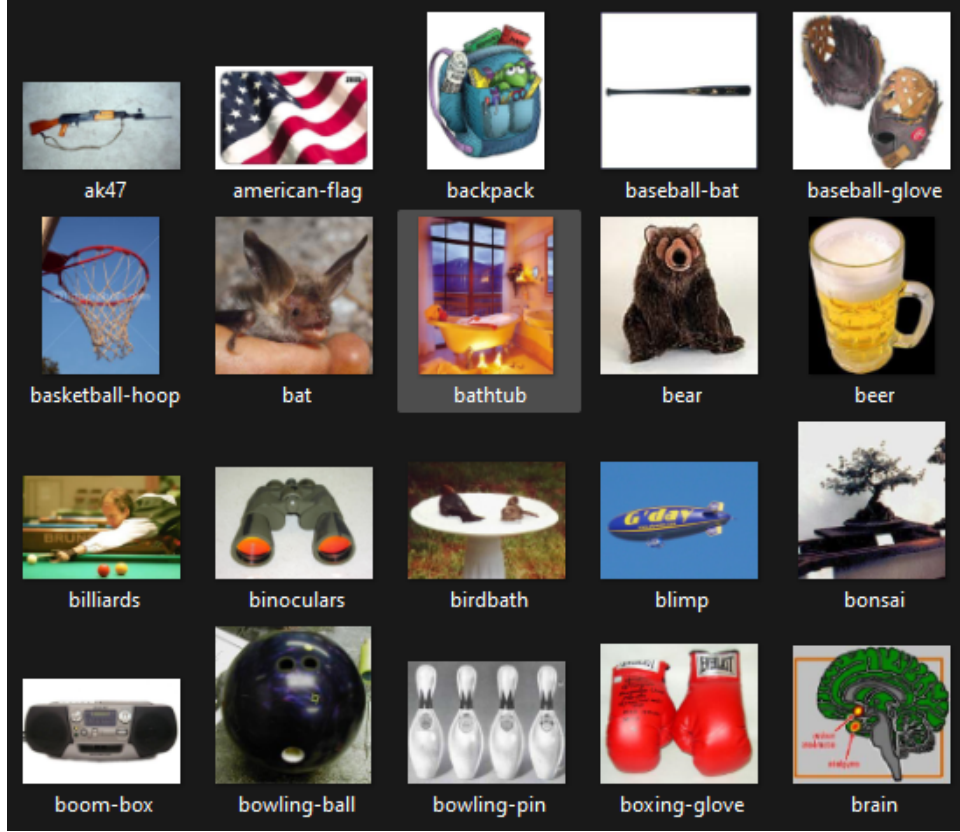


Figure 2.7: *Caltech 20 dataset*

2.4 CNN architecture

This study utilized the VGG16 convolutional neural network, which stands for “Visual Geometry Group 16.” Introduced by Simonyan and Zisserman¹⁸ from the University of Oxford in their 2014 paper “Very Deep Convolutional Networks for Large-Scale Image Recognition,” this model achieved an impressive accuracy of 92.77% on the ImageNet dataset that contained 14 million images falling under 1000 classes.

The “16” in VGG16 refers to the network’s 16 weight layers, which are the layers that have learnable parameters. Specifically, VGG16 comprises 13 convolutional layers, five max-pooling layers, and three fully connected (dense) layers, adding up to a total of 21 layers, but only 16 of these layers have weights (see Figure 2.8). The network takes as input images of size 224×224 with three RGB channels.

An impressive fact about the VGG16 is that its architecture is as simple as it gets: a

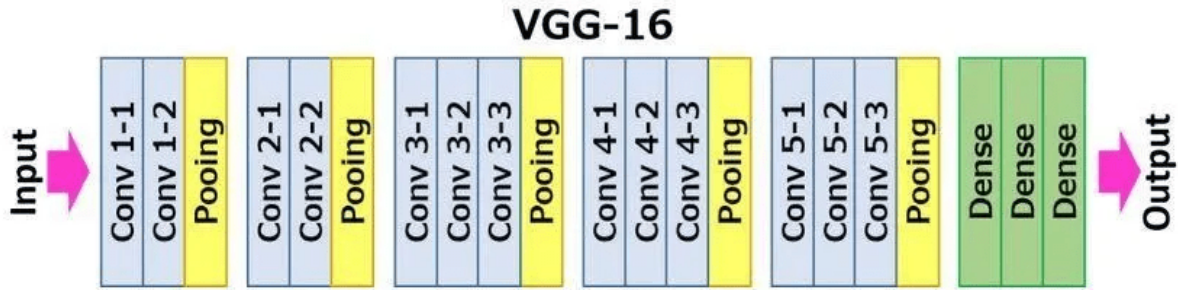


Figure 2.8: *VGG16 architecture map*

bunch of convolutional layers with 3×3 filters and stride 1, while the padding is always the same, coupled with max-pooling layers with 2×2 filters and stride 2. This consistent reliance on small filters and pooling layers throughout the network is a key characteristic of VGG16. As illustrated in Figure 2.9, Conv-1 has 64 filters, Conv-2 has 128, Conv-3 has 256, and both Conv-4 and Conv-5 have 512 filters each. After these convolutional layers, the network includes three fully connected layers: the first two have 4096 channels each, and the third layer, which is designed for classification, has several channels matching the number of classes in the dataset.

The network primarily uses ReLU, short for rectified linear unit as the activation function, except in the final layer where a softmax function is used to produce probabilities for each class. An activation function is a mathematical function that introduces non-linearity allowing the network to learn complicated patterns and mappings. Without it, the network would just be a linear function which is a polynomial of degree one¹⁹.

The optimizer utilized here is RMSprop, which stands for Root Mean Square Propagation. The learning rate here was taken as 0.00001. Optimizers are functions that change neural network attributes: weights and learning rates to decrease the total loss and increase accuracy. A comprehensive comparison study of the most used optimizers in neural networks has been conducted by Desai²⁰.

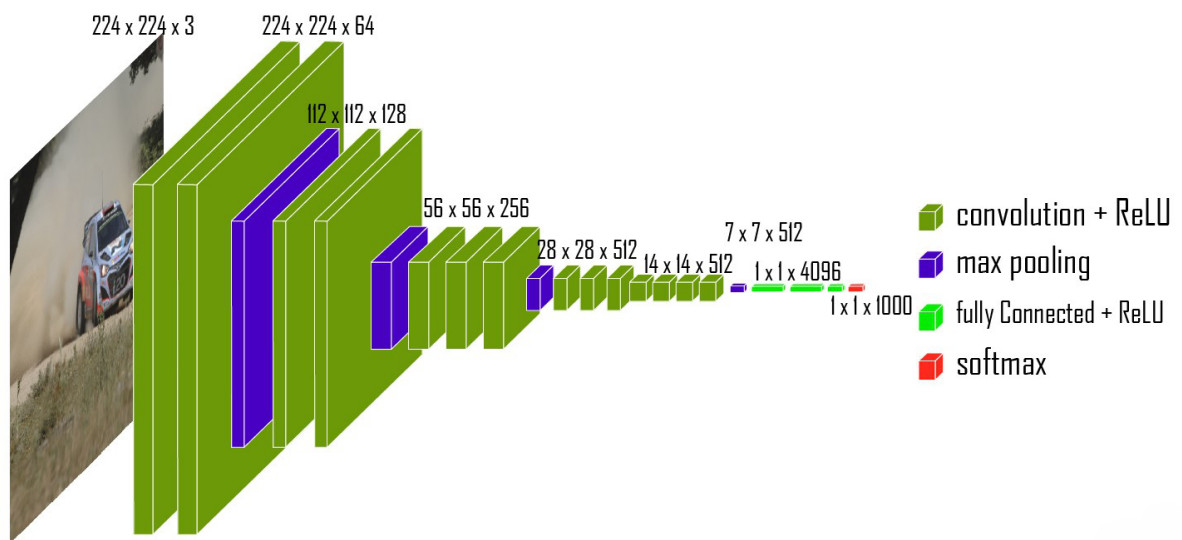


Figure 2.9: *VGG16 architecture*

Chapter 3

Identifying bias using background segments and image scrambling

3.1 Introduction

What is bias?

In the context of CNN image classification, bias is a case of systematic inaccuracies in model predictions due to certain patterns or irregularities within either data or the learning process of a model. This kind of bias would finally affect the way CNN learns to recognize classes and finally result in prejudiced or incorrect classifications.

How are biases caused?

Biases can emerge as early as the data acquisition stage and can be influenced by various factors, which are elaborated on in the following discussion:

1. **Capture Bias:** Despite the best efforts of their creators, the datasets can have a strong built-in bias. For example, datasets captured in a controlled environment, where all images are taken during a single session, may exhibit inherent biases. These biases might be due to consistent artifacts or characteristics from the session rather than from

the objects themselves. A prominent type of bias is that objects are almost always positioned in the center of images. If you search for "mug" on Google Image Search, you will notice another kind of capture bias: nearly all the mugs feature a handle that is oriented to the right. Subtle variations such as changes in lighting conditions or camera temperature can create artifacts that are hard to notice visually but will strongly influence CNN's learning process.

2. **Selection Bias:** This type has been particularly prevalent in World Wide Web-downloaded datasets, where photo acquisition is uncontrolled. Indeed, using only an internet search often fails to provide a representative sample because the keyword-based search returns only certain types of images.²¹ In order to reduce selection bias, it helps to collect data by combining sources: for example, taking images using multiple search engines from different countries.⁹
3. **Irrelevant Features:** CNNs may learn from the background or artifacts in the images rather than focusing on the subject of interest itself. For example, if all images of a dog are presented against a specific background, then the model might learn to associate the background with the dog, rather than recognizing the dog itself.
4. **Imbalanced Datasets:** This is when some classes are dominating while others are not. CNN could be biased towards those classes that occur more frequently. This leads to its poor performance in less common classes, and it could not generalize well for all scenarios.

How do biases impact the classification?

1. **Impacted Decision-Making:** Dataset biases have a substantial impact on the neural network's learning and decision-making. As a result, the model may end up focusing on artifacts, background noise, or biases introduced during the data acquisition process, causing the main object of interest to become secondary or ignored. As a result, the model may base its classifications on these biases rather than the primary object.

2. **Limited Generalization:** Models might excel with training data but struggle with real-world data or diverse test cases because they depend on biased features.
3. **Unreliable Classification:** Even with high classification accuracy, predictions can be unreliable if the CNN focuses on areas that are not the primary object of interest. Bias can also lead to unequal performance across different classes, resulting in unfair predictions.

Therefore, it becomes crucial to identify any classification or dataset biases and be cautious about interpreting the classification results.

How to identify bias?

There are several methods to identify if your dataset has a consistent bias that is influencing CNN's classification:

1. **Cropping Background Segments:** This method involves cropping a small, typically blank section of the background that lacks visual information about the object of interest and then classifying the CNN based on this cropped dataset. If the CNN can accurately predict the class of the image from just this small background portion, it indicates the presence of consistent bias in the dataset, explained in greater detail in section [3.2.1](#).
2. **Image Scrambling:** When the background is inseparable from the foreground and cropping is not feasible, the scrambling method is highly effective. This technique involves dividing the image into small tiles, which are then randomly shuffled, disrupting the image's patterns. Details of this method are discussed in section [3.2.2](#).
3. **Image Transforms:** Applying image transformations like Fourier, Wavelet, and Median filters during the pre-processing stage can smooth out datasets and enhance model robustness. The Wavelet transform, in particular, handles natural and synthetic datasets differently, offering a method for detecting biases. This process is explained in detail in chapter [4](#).

4. **Class Activation Mappings (CAM)**¹¹: Generates heatmaps to show which parts of the input image were most influential in the network’s decision-making. By mapping CNN activations back to the original image, it highlights the regions that most significantly impacted the model’s classification.
5. **Performance Evaluation**: Analyze the confusion matrix to identify patterns of misclassification and whether certain classes contain consistent bias. Additionally, test the CNN on different subsets of data or under varied conditions to see if performance varies significantly.

3.2 Methods

In this chapter, we aim to identify any persistent biases in datasets through two efficient techniques, cropping background segments 3.2.1 and image scrambling 3.2.2. These methods will help us investigate whether CNN is learning primarily from the object of interest or from irrelevant background features. This is crucial for assessing the reliability of the classification accuracy reported by the model. A high classification accuracy might be misleading if the CNN is relying on the bias or background patterns or other non-essential features rather than the core characteristics of the object.

We encourage everyone to follow these methods outlined in this thesis to analyze your dataset for any such biases or background noise or misleading features that CNN might be picking up.

3.2.1 Cropping background segments

Dhar and Shamir¹⁷ have proposed a method that entails extracting a small segment from the background of an image, specifically a 20×20 pixel area from the top-left corner that does not include the object of interest, as shown in Figure 3.1. These background segments are usually blank and devoid of any significant visual details. By compiling a dataset from these cropped, blank background portions and training a CNN to classify them, we can determine

if the model is subconsciously learning from irrelevant background patterns rather than focusing solely on the object itself. Ideally, background features should have no influence on the classification process.

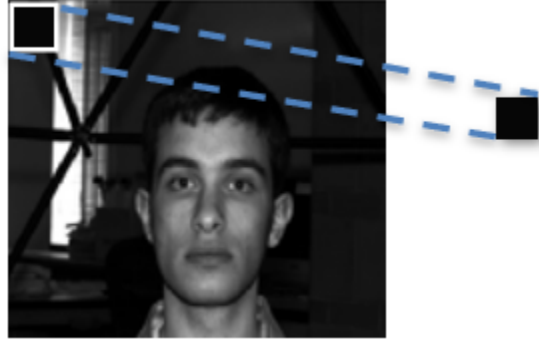


Figure 3.1: 20×20 cropped background segment

Figure 3.2 shows the original Yale Faces²² dataset alongside the newly generated 20×20 cropped dataset. A CNN has been trained and tested on both the original and the cropped datasets, and the intriguing results are discussed in section 3.3.2.

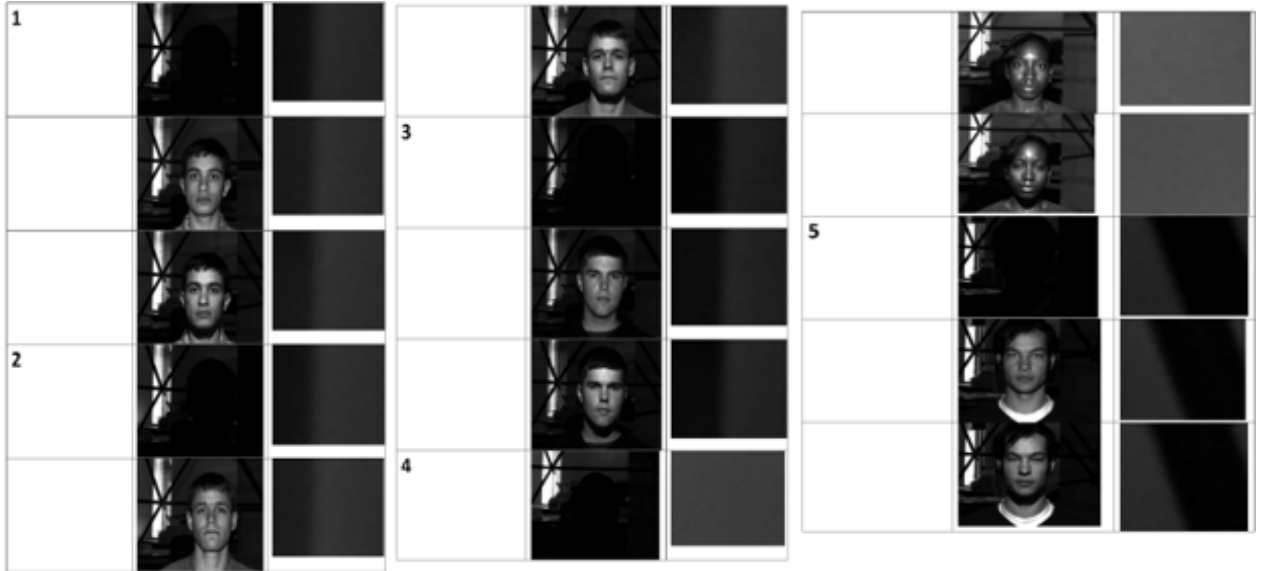


Figure 3.2: 20×20 cropped yale faces¹⁵ dataset

The drawback of this approach is twofold: firstly, a blank background may not always be readily available, making it hard to separate foreground information. It is not feasible when the entire image is a subject of interest and cropping a part of it is not an option. Figure

3.3 shows examples of images where this method would be ineffective due to the lack of a background.



Figure 3.3: *Images with no background*

Secondly, using a blank background might lead to confusion with contextual learning. These limitations can restrict the flexibility and feasibility of implementing the solution. However, these problems can be addressed through the use of the image scrambling technique described in section 3.2.2.

3.2.2 Image scrambling

Image scrambling entails the arbitrary rearrangement of pixels to render the image unrecognizable and disrupt the correlation among the adjacent pixels. It is important to note that while the pixel values themselves remain unchanged during this process, their positions are altered. Image pixels are highly correlated with neighboring pixels and a good scrambling algorithm reduces this correlation near to zero. Mondal²³, Modak and Pawar²⁴ discusses a range of scrambling techniques, such as matrix transformation, bit scrambling, pixel swapping, row and column swapping, Rubik’s cubic algorithm, and Arnold’s transformation.

In this research, a very straightforward technique called “Tile scrambling” was used. This method involves dividing an image into smaller, non-overlapping tiles and then shuffling these tiles in a random order. In order to assess the effectiveness of scrambling, many tile sizes were tested as shown in Figure 3.4. This approach allows for the assessment of how well the model can handle changes in image structure and disrupt patterns formed by neighboring pixels.

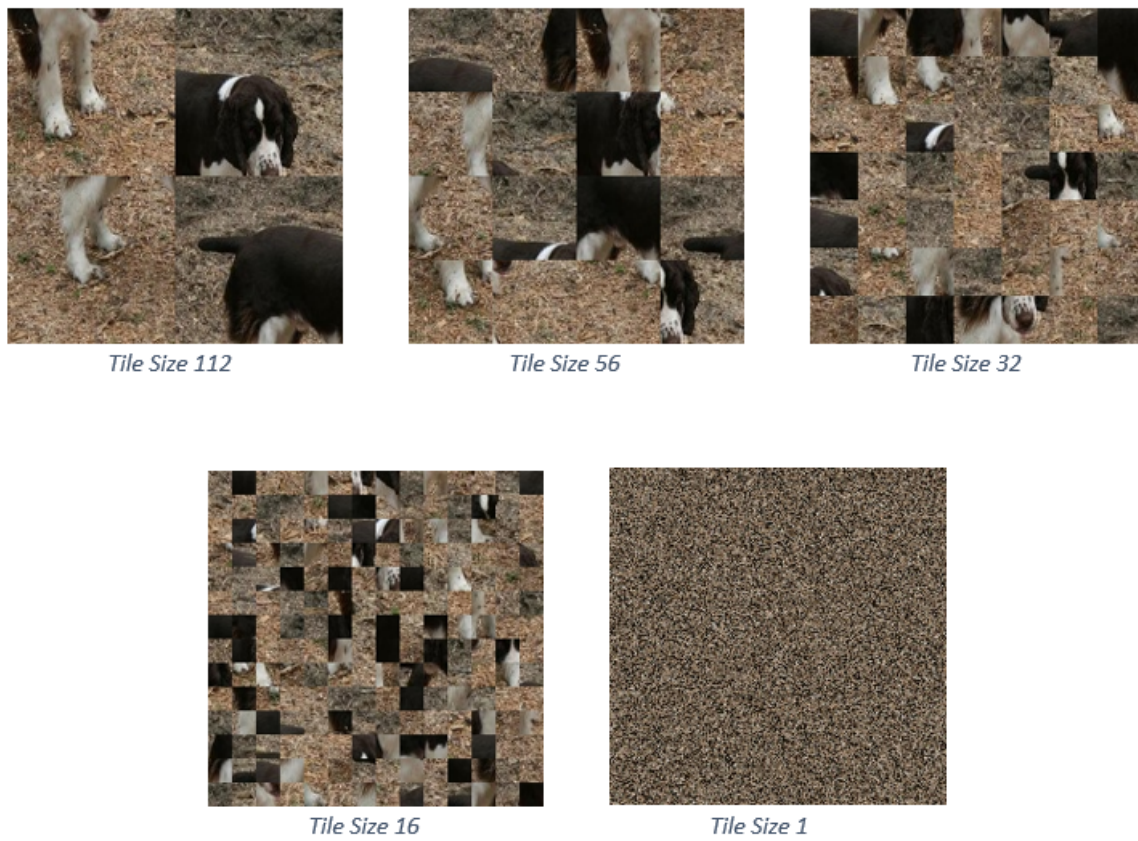


Figure 3.4: *Image scrambling technique outputs*

When an image lacks a clear background or when cropping out a section of the background is not feasible, image scrambling is highly recommended as a technique to assess dataset bias and evaluate CNN performance. A CNN has been trained and tested using this scrambled dataset, and the results of this process are discussed in section [3.3.3](#).

3.3 Results

3.3.1 Classification results

VGG16 models were trained and evaluated on all six datasets described in section 2 without any pre-processing. Table 3.1 presents the classification results for the models tested on the raw datasets. The results show that our model accurately predicts the classes and achieves high classification accuracy across all six datasets. Notably, the model performed exceptionally well with the Coil-20 dataset, achieving an accuracy of $\sim 100\%$, which raises concerns about the model possibly memorizing dataset-specific biases due to Coil-20 being synthetically generated. On the other hand, the Caltech 20, which features a combination of natural and non-natural images, showed a lower accuracy of around $\sim 36.6\%$, reflecting the increased variability and complexity of this dataset compared to others.

Dataset	Classification accuracy	Random accuracy
Imagenette	59%	10%
Natural images	85%	14.28%
CT scans	70%	25%
Coil-20	100%	5%
Yale Faces	70%	6.66%
Caltech 20	36.6%	5%

Table 3.1: *Classification accuracies of raw datasets*

In the subsequent sections, we will investigate whether these classification accuracies exhibit any biases or if background patterns are influencing the model’s learning process. This analysis will help us assess the reliability of these accuracies and determine the trustworthiness of our classification model.

3.3.2 Cropped background segments results

The cropping background segments method discussed in Section 3.2.1 has been applied to all six datasets. The output of applying this method to the Imagenette dataset (Figure 2.2), is depicted in Figure 3.5. The cropped images lack any features related to the object, are highly similar, and are indistinguishable from the naked eye.



Figure 3.5: *Corresponding 20×20 cropped Imagenette dataset*

A VGG16 model detailed in Section 2.4 was trained and tested on all the datasets, including both the original and cropped versions. Table 3.2 presents the classification accuracies when trained on the cropped datasets, compared to those trained on the original datasets.

Dataset	Full image accuracy	Cropped accuracy	Random accuracy
Imagenette	59%	17%	10%
Natural images	85%	27.6%	14.28%
CT scans	70%	44%	25%
Coil-20	100%	27.8%	5%
Yale Faces	70%	17.7%	6.66%
Caltech 20	36.6%	12.7%	5%

Table 3.2: *Classification accuracies of cropped datasets*

It is evident from the results that models trained and tested on 20×20 pixel cropped datasets achieved much higher classification accuracy than random guessing in all cases. The accuracies for cropped images are consistently at least twice as high as the random accuracies across all datasets (see Figure 3.6). This suggests that the CNN can accurately

predict the object’s class by examining the top left 20×20 pixel section of the image, which lacks the object’s features and is quite uniform across different images.

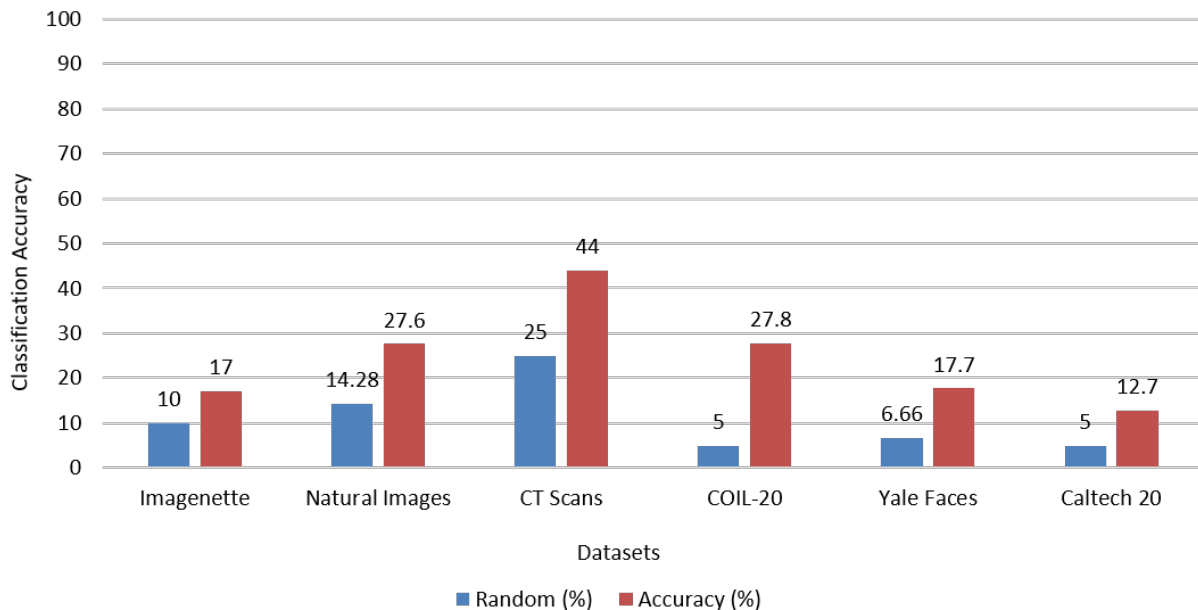


Figure 3.6: *Classification accuracies of cropped segments vs random chance*

The results reveal that the Coil-20 dataset, which was synthetically produced in a controlled environment show significantly higher classification accuracy compared to datasets gathered from diverse sources where images were not specifically intended for object recognition tasks. The random guessing accuracy of 5% was improved by nearly 550%, reaching $\sim 27.8\%$ when using just a 20×20 pixel region from each original image. Similarly, the Yale Faces dataset with an initial random guessing accuracy of 6.66%, saw an increase of about 260%, rising to around $\sim 17.7\%$. These findings suggest that image features that are irrelevant for object recognition but help distinguish between image classes are more prevalent in non-natural datasets, while such features are less dominant in datasets collected from a variety of sources.

A possible explanation for such high cropped accuracy is that the background contains a strong signal, which significantly influences and creates a consistent bias in the network’s learning process. This bias, though not directly related to the object’s features, can distinguish between image classes, resulting in better-than-expected performance.

3.3.3 Image scrambling results

The image scrambling technique outlined in Section 3.2.2 was applied to all six datasets. Three tile sizes were used in this experiment: 1, 16, and 32. With tile sizes of 1 and 16, the scrambled images are virtually unrecognizable to the human eye and difficult to classify. Tile size 32 offers some level of guess-ability, though it is not consistently reliable. Figure 3.4 provides a clear comparison of the effects of each tile size.

For instance, Figure 3.7 depicts the scrambling results of the imagenette dataset (Figure 2.2), where the tiles are 16x16 pixels.

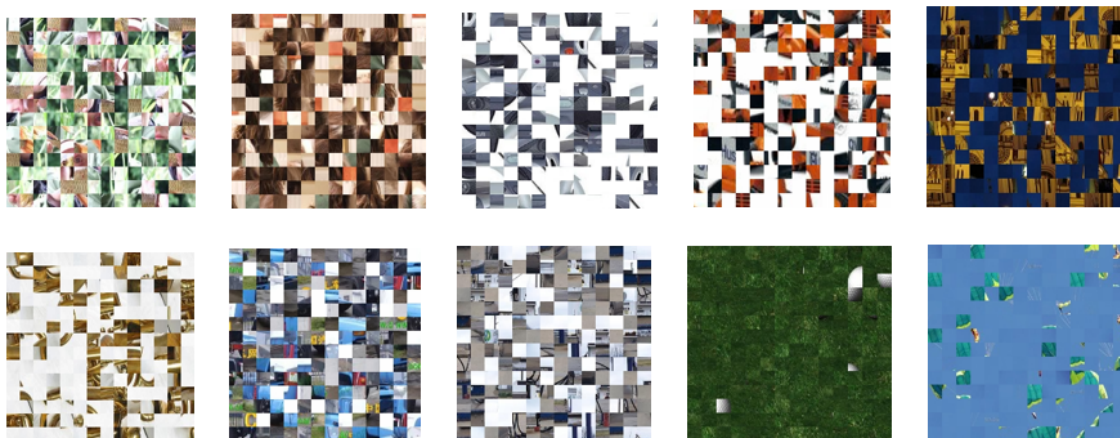


Figure 3.7: *Corresponding scrambled Imagenette dataset with tile size 16*

A VGG16 model detailed in Section 2.4 was trained and tested on all six datasets, including both the original and scrambled versions. Table 3.3 presents the classification accuracies for each model when trained on the scrambled datasets, compared to those trained on the original datasets.

A consistent pattern is observed across all datasets and tile sizes: models trained on scrambled datasets still achieve much higher accuracy than random chance. Let us take a closer look at the results of a tile size of 1, when using a tile size of 1, where each pixel in an image is rearranged randomly, the image loses all recognizable structure and patterns, making it nearly impossible for humans to identify the class. Surprisingly, despite this, our VGG16 model manages to classify these scrambled images with unexpectedly high accuracy rates.

Dataset	Original accuracy	tile size = 1	tile size = 16	tile size = 32	Random
Imagenette	59%	10%	28.4%	30.3%	10%
Natural images	85%	40.47%	36.3%	38.78%	14.28%
CT scans	70%	38%	44%	42%	25%
Coil-20	100%	20%	13%	10%	5%
Yale Faces	70%	18.8%	9%	13.3%	6.66%
Caltech 20	36.6%	11.6%	14.1%	8.1%	5%

Table 3.3: *Classification accuracies of scrambled datasets*

The Coil-20 dataset exhibits the highest bias or increase in accuracy compared to random guessing. Its random guessing accuracy of 5% was improved by 400%, reaching $\sim 20\%$. The Yale Faces dataset shows the second highest increase, with accuracy rising from a random guessing rate of 6.6% to $\sim 18.8\%$, marking an increase of about 285%. Other datasets also showed a high classification accuracy for scrambled images. In contrast, the Imagenette dataset displayed the least bias, with the scrambled accuracy matching the random guessing accuracy for a tile size of 1, which aligns with the expected behavior when images are scrambled.

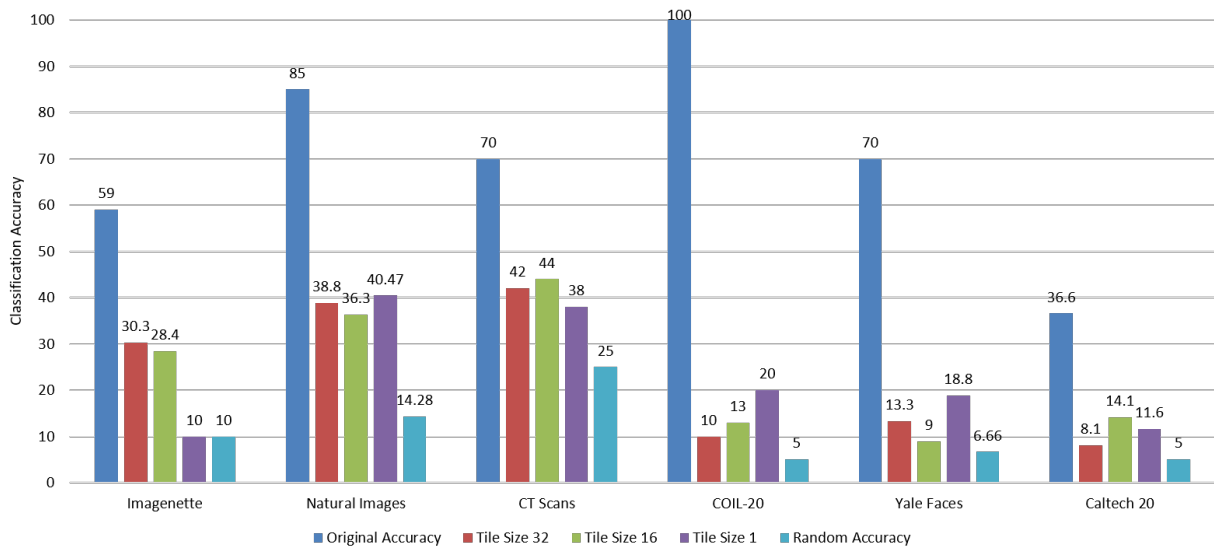


Figure 3.8: *Classification accuracies of scrambled datasets vs random chance*

In conclusion, our VGG16 models were able to accurately classify scrambled images, which would be impossible for humans to identify, as shown in Figure 3.7. This indicates that nearly all datasets contain some level of bias, with the CNN memorizing irrelevant patterns and being influenced by these biases. Additionally, synthetic datasets such as Coil-20 and Yale Faces exhibited the highest level of bias compared to others.

3.4 Conclusion

We have analyzed the classification results for both methods: Cropping background segments (section 3.2.1) and Image scrambling (section 3.2.2). Both methods produced significantly higher accuracies compared to random chance.

A 20×20 pixel background segment, which typically lacks object features and is usually blank, should ideally not contribute to the neural network’s learning and decision-making process. However, this is not the case here. The CNN achieved significantly higher accuracies than random chance (see Table 3.2), suggesting that these 20×20 pixel cropped segments contain information not visible to the naked eye, which helps the model in distinguishing between image classes. Consequently, CNN appears to be learning from this hidden bias and relying on it for its final predictions.

When images are divided into tiles and scrambled randomly, predicting the object in the image becomes virtually impossible. Nonetheless, the model still achieves accuracy that is twice as high as random guessing, and in some cases, even higher (refer to Table 3.3). This suggests that the model may be memorizing specific patterns and that a consistent bias is influencing its classification.

Results clearly show that these methods are highly effective in revealing dataset biases and any significant hidden background information that is influencing CNN’s predictions. By applying these techniques, one can gain insight into the reliability of CNN classification accuracy and approach it with greater caution.

Chapter 4

Identifying bias using image transforms

In Chapter 3, two techniques were discussed: cropped background segments and tile scrambling. These methods were effective in detecting general dataset biases. However, they struggled to differentiate between biases present in natural versus synthetic datasets.

Synthetic datasets are generated in a controlled environment, which theoretically reduces the likelihood of bias since all parameters are regulated. Despite this, biases often still occur. For instance, if all images of an object are captured in a single session, a strong and consistent bias can be introduced at the image acquisition stage itself. This capture bias is challenging to detect and can affect the CNN’s classification accuracy. Consequently, the classification will be performed on session bias rather than focusing solely on the object of interest.

In contrast, natural datasets often face different challenges. Selection bias is particularly prevalent because images are typically collected from various sources. The methods used for filtering and selecting data can introduce selection bias, which in turn may result in representation bias by failing to fully capture the diversity or complexity of real-world scenarios.

In this chapter, we will aim to identify a method that effectively detects and differentiates between contextual information and the presence of background noise. We will apply a range

of transformations, such as Fourier and Wavelet transforms, along with Median filtering, to the datasets. Following these pre-processing steps, we will train a CNN and evaluate how these transformations impact classification accuracy.

4.1 Methods

4.1.1 Fourier transform

The Fourier transform is one of the most widely employed methods in image processing to break down an image into its sine and cosine components. The transformation's outcome depicts the image in the frequency domain, whereas the input image corresponds to the spatial domain. In the resulting image, each point denotes a certain frequency present in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction, and image compression.

As we are only concerned with digital images, we will restrict this discussion to the Discrete Fourier Transform (DFT). Since DFT is the sampled Fourier Transform, it does not contain all frequencies forming an image, but only a set of samples that is good enough to fully describe the image in the spatial domain. The number of frequencies equals the number of pixels in the spatial domain image, i.e. the image in the spatial and Fourier domains are of the same size²⁵.



Figure 4.1: *Fourier transformed image*

As per OpenCV’s documentation²⁶, the initial step involved converting all images to grayscale followed by applying the discrete Fourier transform. The resulting image in the Fourier domain is shown in Figure 4.1. Along with the full images, the 20×20 pixel cropped images were also transformed into Fourier domain, shown in Figure 4.2.

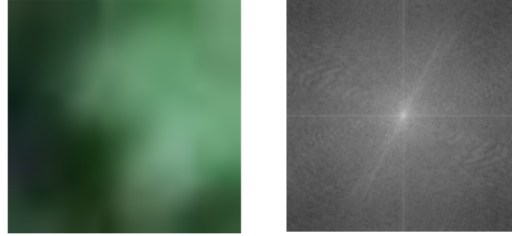


Figure 4.2: *Fourier transform of 20×20 cropped image*

Results

A VGG16 model has been trained and tested on the Fourier-transformed datasets and the classification accuracies are shown in Table 4.1.

Dataset	Full images		Cropped images		Random
	Original	Fourier	Original	Fourier	
Imagenette	59%	38%	17%	10%	10%
Natural images	85%	76.8%	27.6%	22%	14.28%
CT scans	70%	54%	44%	40%	25%
Coil-20	100%	80%	27.8%	26%	5%
Yale Faces	70%	60%	17.7%	13%	6.66%
Caltech 20	36.6%	30%	12.7%	12%	5%

Table 4.1: *Classification accuracies of Fourier transformed datasets*

A consistent trend is noticeable in the results: applying the Fourier transform has reduced the classification accuracy across all the datasets, both for full images (see Figure 4.3) and 20×20 cropped images (see Figure 4.4). For example, in the Imagenette dataset, the accuracy

dropped significantly from $\sim 59\%$ to $\sim 38\%$ for full images, and the accuracy for cropped images fell to random levels. Similarly, the Coil-20 dataset saw its accuracy decline from $\sim 100\%$ to $\sim 80\%$. Not just these two datasets, but nearly all datasets experience a decrease in classification accuracy for both full and cropped images.

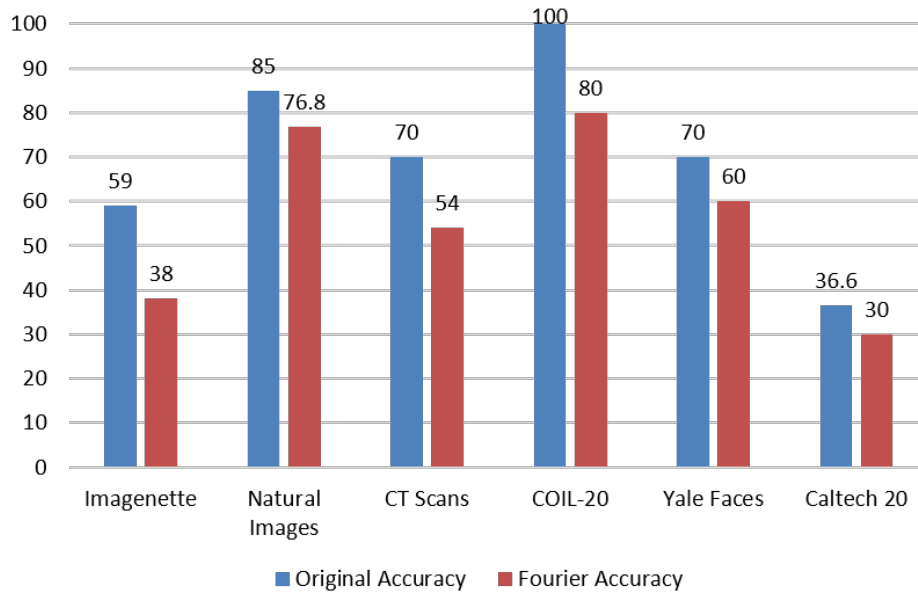


Figure 4.3: *Classification accuracies of Fourier transformed full images*

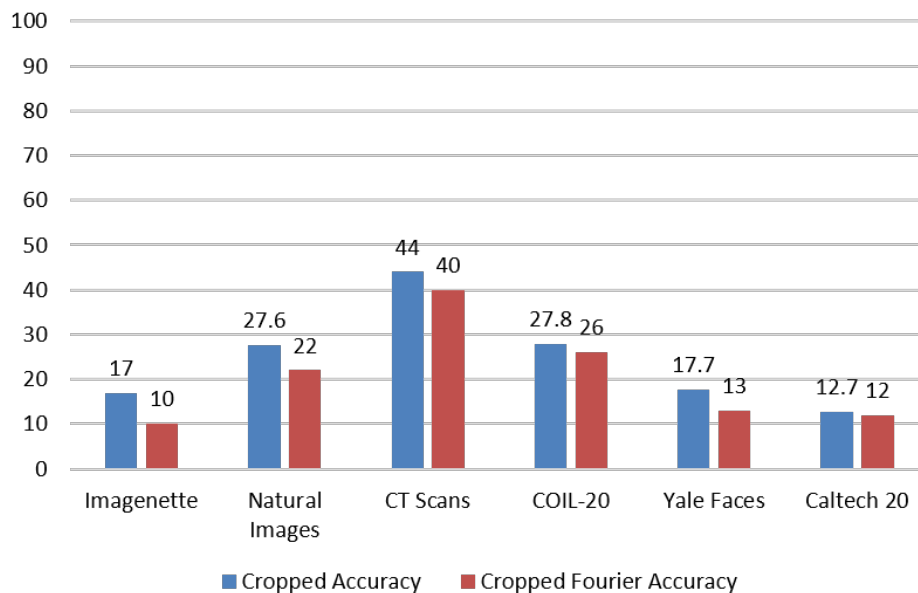


Figure 4.4: *Classification accuracies of Fourier transformed cropped images*

One possible explanation for this decrease could be that transforming images into the Fourier domain leads to a loss of spatial information. CNNs rely on this spatial information to effectively learn and classify features. It gets challenging to determine whether the drop in accuracy is due to the loss of relevant features or from the reduction of inherent bias in the datasets that were influencing CNN’s predictions.

However, the Fourier transform does not effectively help in distinguishing between biases in natural and non-natural datasets, as it reduces accuracy across all types of datasets.

4.1.2 Wavelet transform

Wavelet transforms are mathematical methods used for investigating and retrieving data from images.^{27;28} It breaks an image into components at various positions, preserving both frequency and spatial data. In contrast to the Fourier Transform, which outputs a global frequency, the Wavelet Transform provides a time-frequency (or space-frequency) representation, allowing for localized evaluation in both space and frequency domains. Wavelet transforms are presently being implemented to replace Fourier transforms in a wide range of applications, including medical imaging, image retrieval, watermarking, and many more.

Discrete Wavelet Transform (DWT) is applied to all our datasets, full images along with the 20×20 cropped ones, using a python package named PyWavelets²⁹. Two different wavelets have been used in this study: Haar and Daubechies. Haar wavelets are discontinuous in nature and seem like a step function, while Daubechies wavelets best represent polynomial trends. Figure 4.5 shows the images produced by applying these two wavelets, that appear quite comparable to one another.

Results

A VGG16 model was trained and evaluated on datasets transformed using the Discrete Wavelet Transform, including both full and cropped images. Table 4.2 shows the classification accuracies for models trained on Haar wavelet-transformed data.

The findings indicate a notable decline in prediction accuracies for natural datasets, while



Figure 4.5: *Discrete Wavelet transformed image*

Dataset	Full images		Cropped images		Random
	Original	Haar	Original	Haar	
Imagenette	59%	50%	17%	13%	10%
Natural images	85%	72.77%	27.6%	24.5%	14.28%
CT scans	70%	69%	44%	56%	25%
Coil-20	100%	100%	27.8%	31.4%	5%
Yale Faces	70%	80%	17.7%	16%	6.66%
Caltech 20	36.6%	35%	12.7%	12.5%	5%

Table 4.2: *Classification accuracies of Haar Wavelet transformed datasets*

non-natural datasets have seen either consistent or improved accuracy rates. This trend is observed in both full images and cropped images. For instance, as shown in Figure 4.6 the prediction accuracy for the Imagenette dataset has dropped from $\sim 59\%$ to around $\sim 50\%$ for full images, and from $\sim 17\%$ to roughly $\sim 13\%$ (nearly random levels) for cropped images. In contrast, the Coil-20 dataset, which is synthetic, accuracy maintains a constant accuracy of 100%, and its cropped image accuracy has increased from $\sim 27.8\%$ to about $\sim 31.4\%$.

The 20×20 cropped images are usually blank and do not contain any visual information, but they still contain hidden information that can influence a CNN model’s predictions. Applying a wavelet transform to these cropped images has increased their accuracy. This indicates that the Wavelet transform successfully revealed and enhanced the hidden signals in the background. Consequently, the bias in these non-natural datasets has risen. As a

result, CNN has adapted to this increased bias and has become less reliable.

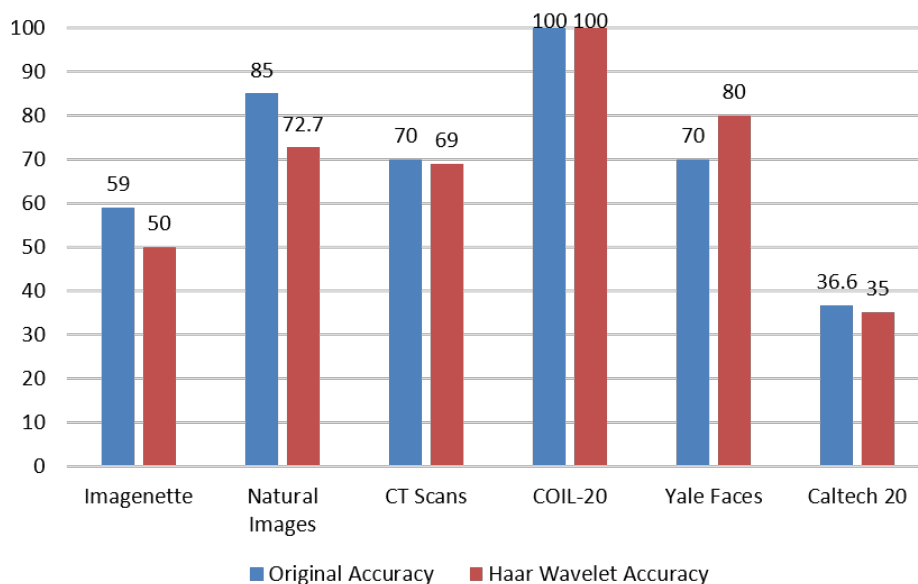


Figure 4.6: *Classification accuracies of Wavelet transformed full images*

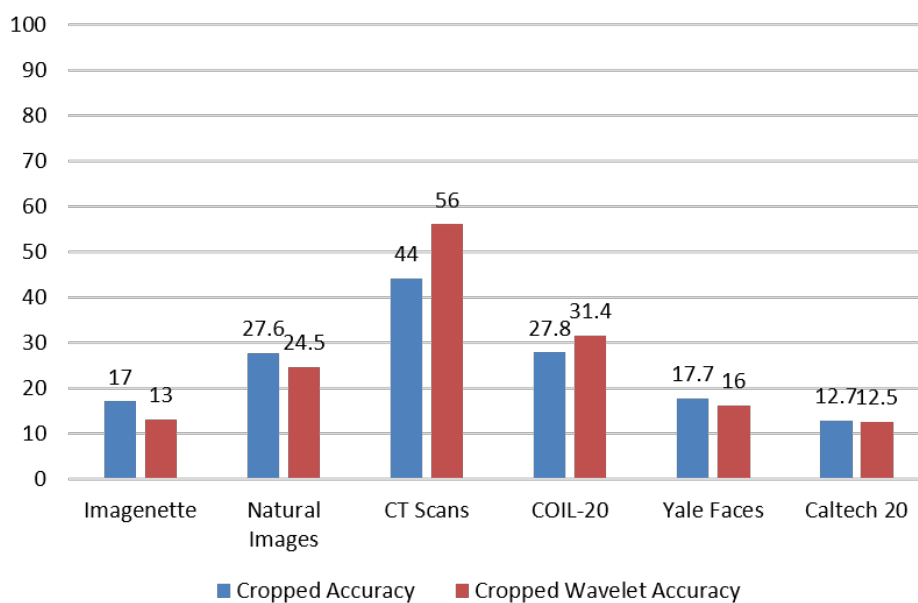


Figure 4.7: *Classification accuracies of Wavelet transformed cropped images*

In contrast, natural datasets typically do not contain capture bias and are therefore affected differently by the wavelet transform. As a result, the accuracies for Imagenette and Natural image datasets do not improve. We have now discovered a transform that

effectively differentiates between contextual information and the presence of background noise and addresses them accordingly.

Table 4.3 presents the classification accuracies for models trained on data transformed using Daubechies wavelets. The performance of Daubechies wavelets is similar to that of Haar wavelets and also effectively differentiates between biases in natural and non-natural datasets.

Dataset	Full images		Cropped images		Random
	Original	Daubechies	Original	Daubechies	
Imagenette	59%	45.5%	17%	13%	10%
Natural images	85%	72%	27.6%	24%	14.28%
CT scans	70%	69%	44%	56%	25%
Coil-20	100%	100%	27.8%	30%	5%
Yale Faces	70%	80%	17.7%	16%	6.66%
Caltech 20	36.6%	37.5%	12.7%	12.7%	5%

Table 4.3: *Classification accuracies of Daubechies Wavelet transformed datasets*

The wavelet transform breaks down an image into numerous magnitudes, capturing features at various levels of detail. This breakdown highlights various frequencies such as approximation, and horizontal, vertical, and diagonal details, which can reveal edges, textures, and patterns. Compared to the Fourier transform, which only presents frequency details and not spatial data, the wavelet transform provides a time-frequency representation, allowing for localized examination in both the spatial and frequency domains. Additionally, the wavelet transform aids in noise reduction by adjusting wavelet coefficients to clean up images before they are fed into a classifier. These capabilities explain why the wavelet transform can effectively differentiate between contextual information and the presence of background noise and address them accordingly.

4.1.3 Median Filter

A median filter is a non-linear image processing method that evaluates the image pixel by pixel and replaces each pixel with the median of adjacent entries. The pattern of nearby pixels depicts a window that slides across the entire image, one entry at a time. The window is usually a box or cross pattern, centered on the pixel being analyzed.³⁰

It is a smoothing technique that effectively removes disturbances from a noisy image or a signal. Unlike low-pass FIR filters, median filters are notable for retaining image edges, making them widely used in processing digital images.

Median filtering was performed on all our datasets using OpenCV’s ‘medianBlur()’³¹ method to smooth the images and reduce noise. Figure 4.8 shows the outcome of this process, highlighting the improved smoothness, reduced noise, and partial removal of watermarks.



Figure 4.8: *Median filtered image*

Results

A CNN was trained and tested on datasets that had undergone median filtering, including both full and 20×20 cropped background images. The classification accuracies are presented in Table 4.4. The results show that the Median filter can distinguish between contextual information and the presence of background noise. For full images, the accuracy of natural datasets has decreased; for example, Imagenette dropped from $\sim 59\%$ to about $\sim 55\%$, and Natural images fell from $\sim 85\%$ to about $\sim 81.6\%$. In contrast, the accuracy for synthetic datasets has remained stable, while hybrid datasets have improved. For instance, the

accuracy for Caltech 20 increased from around $\sim 36.6\%$ to around $\sim 39\%$.

Dataset	Full images		Cropped images		Random
	Original	Median	Original	Median	
Imagenette	59%	55%	17%	13.9%	10%
Natural images	85%	81.66%	27.6%	23.5%	14.28%
CT scans	70%	69%	44%	42%	25%
Coil-20	100%	100%	27.8%	25%	5%
Yale Faces	70%	70%	17.7%	17%	6.66%
Caltech 20	36.6%	39%	12.7%	12.2%	5%

Table 4.4: *Classification accuracies of Median filtered datasets*

The median filter smooths the image and reduces noise, revealing that natural datasets had a visual bias that was eliminated by the filter. In contrast, synthetic datasets contain hidden information that the median filter cannot remove but can help reveal. As a result, the median filter addresses biases in natural and non-natural datasets differently.

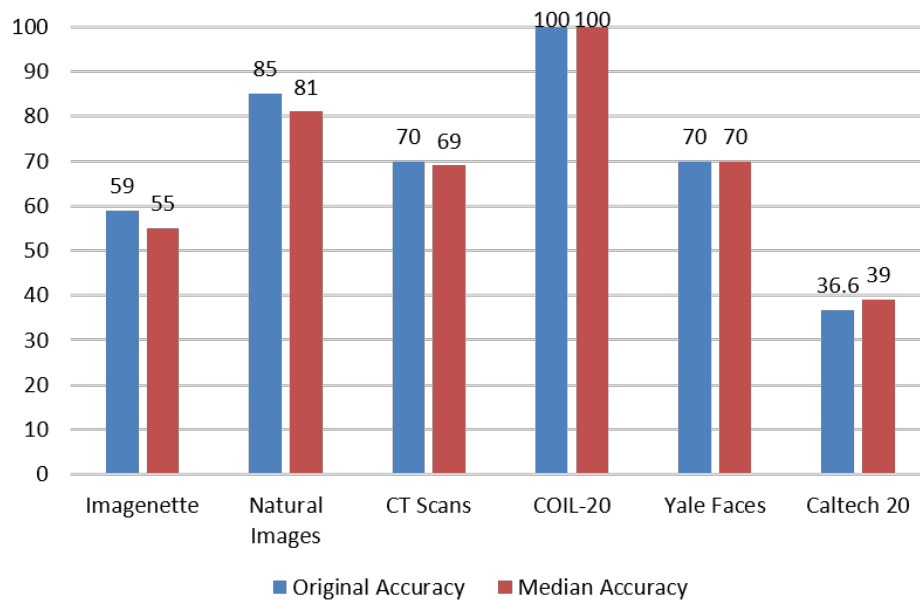


Figure 4.9: *Classification accuracies of Median filtered full images*

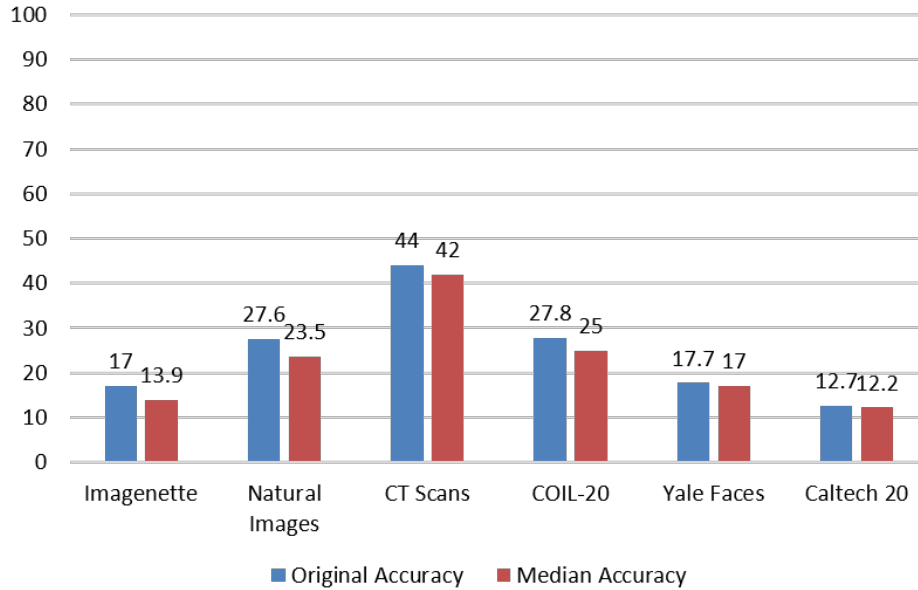


Figure 4.10: *Classification accuracies of Median filtered cropped images*

Since median filtering has proven to be effective and yielded promising results, we will now experiment by applying Fourier and Wavelet transforms on the median-filtered datasets. This will be covered in the subsequent sections.

4.1.4 Median filter + Fourier transform

Now, let's experiment by combining Median filtering and Fourier transform and examine see how it impact classification accuracies. First, median filtering is applied to the images to reduce noise and smooth them out. Next, the Fourier transform is applied to the median-filtered images and are converted into the Fourier domain. Figure 4.11 displays the resultant image after applying the median filter and the Fourier transform.

Table 4.5 shows the classification accuracies of VGG16 models trained and tested on the datasets, including both full images and 20×20 cropped background images that have been median filtered and transformed into Fourier domain. We have observed a substantial and consistent decrease in accuracies across all datasets. For instance, Yale Faces and Imagenette are among those with the most significant reductions in accuracy.

Earlier, we observed that converting images to the Fourier domain results in some loss

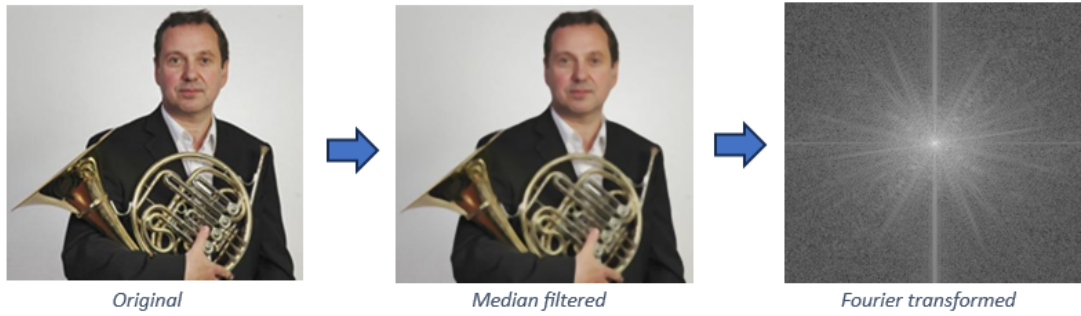


Figure 4.11: *Median + Fourier transformed image*

Dataset	Full images		Cropped images		Random
	Original	Median+Fourier	Original	Median+Fourier	
Imagenette	59%	28%	17%	10%	10%
Natural images	85%	78.8%	27.6%	16%	14.28%
CT scans	70%	71.4%	44%	39.7%	25%
Coil-20	100%	58%	27.8%	26%	5%
Yale Faces	70%	6.6%	17.7%	13%	6.66%
Caltech 20	36.6%	30%	12.7%	14%	5%

Table 4.5: *Classification accuracies of Median filtered Fourier transformed datasets*

of spatial information, leading to a decrease in accuracy. Additionally, the median filter smooths the images and reduces noise, which also contributes to a modest decline in accuracy. Combining these two techniques is expected to cause a more significant drop in accuracy.

Since the Fourier transform was not particularly effective at distinguishing between biases in natural and synthetic datasets, combining it with the median filter also proved ineffective. This combination did not differentiate but instead led to a decrease in accuracies across nearly all datasets.

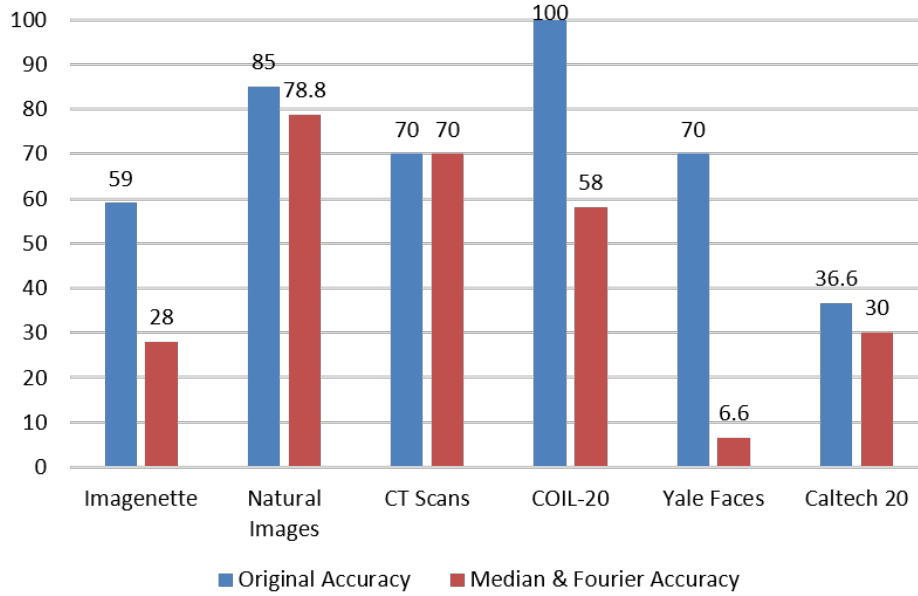


Figure 4.12: *Classification accuracies of Median and Fourier transformed full images*

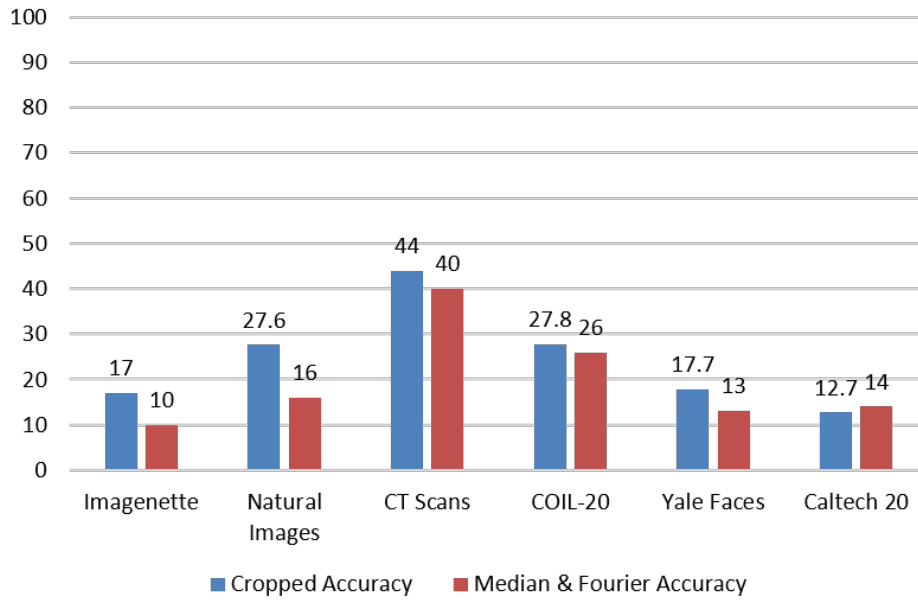


Figure 4.13: *Classification accuracies of Median and Fourier transformed cropped images*

4.1.5 Median filter + Wavelet transform

In this section, we will investigate the effects of combining median filtering with the Haar wavelet transform. First, median filtering was applied to all datasets to smooth the images and reduce noise. Following this, the images were converted into the space-frequency domain

using the Haar wavelet transform. The resulting images, after both median filtering and the Haar wavelet transform, are displayed in Figure 4.14. This approach aims to leverage the benefits of both methods.



Figure 4.14: *Median + Wavelet transformed image*

Table 4.6 presents the classification accuracies for VGG16 models trained and tested on datasets pre-processed with both median filtering and the Haar wavelet transform. We previously observed that median filtering and wavelet transforms were each effective in identifying and distinguishing contextual information and the presence of background noise. Combining these two methods has also proven to be highly successful.

Dataset	Full images		Cropped images		Random
	Original	Median+Wavelet	Original	Median+Wavelet	
Imagenette	59%	49.5%	17%	14%	10%
Natural images	85%	70.2%	27.6%	22.5%	14.28%
CT scans	70%	70.5%	44%	33.4%	25%
Coil-20	100%	100%	27.8%	30%	5%
Yale Faces	70%	80%	17.7%	11%	6.66%
Caltech 20	36.6%	36%	12.7%	13%	5%

Table 4.6: *Classification accuracies of Median filtered Wavelet transformed datasets*

The accuracies for natural datasets, such as Imagenette and Natural images, decreased by about $\sim 9.5\%$ and $\sim 14.8\%$ respectively. While those for synthetic and hybrid datasets either

remained stable or increased. This can be attributed to the median filter, which smooths and reduces noise in the images, and the wavelet transform, which identifies and enhances the capture bias (hidden information present in the background introduced during the image acquisition stage) in synthetic datasets. This pairing accounts for the observed changes in classification accuracies.

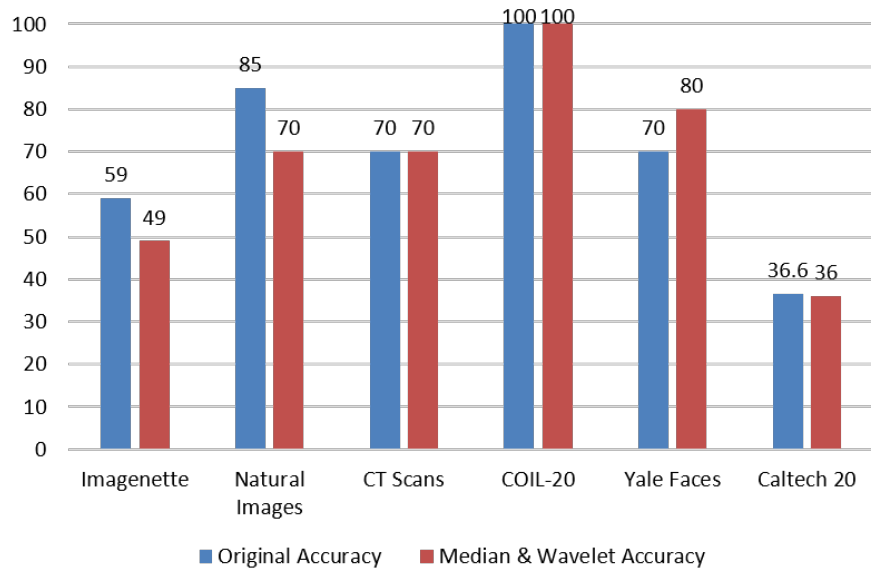


Figure 4.15: *Classification accuracies of Median and Wavelet transformed full images*

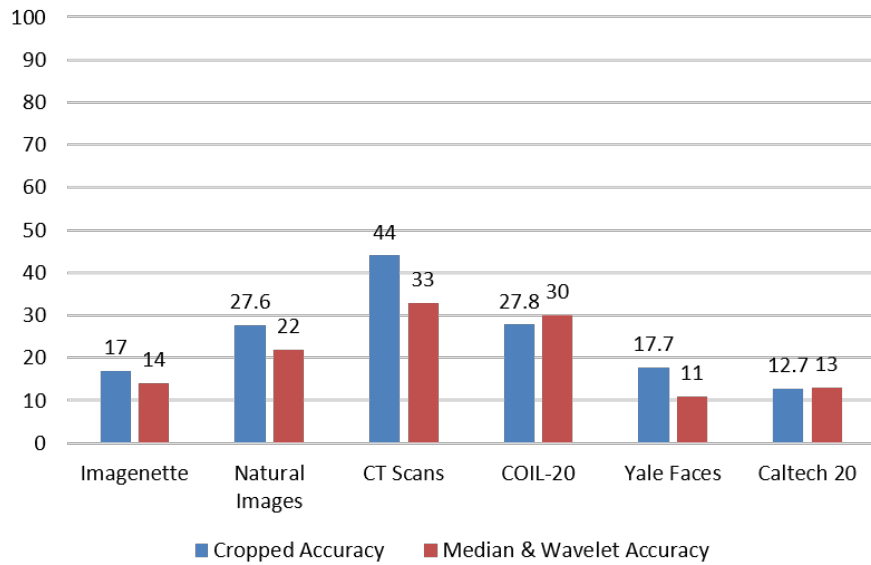


Figure 4.16: *Classification accuracies of Median and Wavelet transformed cropped images*

4.2 Conclusion

In Section 4.1, we reviewed several techniques, including Fourier, Wavelet transforms, Median filtering, and their combinations, which were used to identify bias and distinguish between contextual information and the presence of background noise. The Wavelet transform 4.1.2, Median filtering 4.1.3, and their combination 4.1.5 have all proven highly effective in distinguishing between biases in natural and non-natural datasets. The results show that the method can distinguish between contextual information and the presence of background noise, and alert on the presence of background noise even without the need to use background information.

Applying a median filter smooths the image and eliminates noise, and subsequently applying Wavelet transform to this smoothed dataset further enhances the signal in the image. Notably, the wavelet transform also highlights any hidden signals introduced during the image acquisition process. In a nutshell, these methods decreased the prediction accuracies for natural datasets but improved the accuracies for non-natural datasets, effectively addressing the distinct biases present in each.

Before fully trusting the high prediction accuracies of CNNs, it is strongly recommended to apply the methods outlined in Chapter 3 to determine if there is any persistent bias present in the dataset. Then the techniques discussed in Chapter 4 should help you distinguish between contextual information and the presence of background noise. This evaluation helps gauge the reliability of the CNN and ensures that its accuracy is trustworthy.

4.3 Code repository

The complete code utilized in this research study is available in a GitHub repository: <https://github.com/SaiTeja-Erukude/identifying-bias-in-dnn-classification>.³²

4.3.1 Files description

`vgg16_model.py`: This file contains the architecture of the VGG16 neural network, which was used for model training.

Within the preprocess folder, the following files are included:

- `augment.py`: This script is responsible for augmenting images from the Yale Faces dataset.
- `crop.py`: This file implements the cropping background segments method discussed in Section 3.2.1.
- `scramble.py`: This script executes the “Tile Scrambling” technique, which randomly shuffles the pixels of the images as outlined in Section 3.2.2.
- `fourier.py`: This script applies the Fourier transform to all images located in a specified directory.
- `wavelet.py`: This script applies wavelet transformations to images. It can accommodate various types of wavelets by simply modifying the type specified on line 46.
- `median.py`: This script uses OpenCV’s `medianBlur()` method to apply median filtering with a window size of 5.

Within the predict folder, the following files are included:

- `predict.py`: This script is used for importing the trained model and evaluating its performance on a set of test images.

- `ensemble_predict.py`: This script creates an ensemble classifier that incorporates models trained on the original images, as well as those transformed by Fourier and wavelet techniques.

Bibliography

- [1] Abdellatif Abdelfattah. Image classification using deep neural networks - a beginner friendly approach using tensorflow, 2017. URL <https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using->
- [2] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, and Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *ArXiv*, 2018. URL <https://doi.org/10.48550/arXiv.1803.01164>.
- [3] Keras. Keras resnet documentation, 2024. URL <https://keras.io/api/applications/resnet/>.
- [4] Arden Dertat. Applied deep learning - part 4: Convolutional neural networks, 2017. URL <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>.
- [5] Beilun Wang, Rui Ma, Jingyu Kuang, and Yan Zhang. How decisions are made in brains: Unpack “black box” of cnn with ms. pac-man video game. *IEEE Access*, 8: 142446–142458, 2020. doi: 10.1109/ACCESS.2020.3013645.
- [6] Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey. *Machine Learning and Knowledge Extraction*, 3(4):966–989, 2021. ISSN 2504-4990. doi: 10.3390/make3040048. URL <https://www.mdpi.com/2504-4990/3/4/48>.
- [7] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. *CoRR*, abs/1505.01257, 2015. URL <http://arxiv.org/abs/1505.01257>.

- [8] Ian Model and Lior Shamir. Comparison of data set bias in object recognition benchmarks. *IEEE Access*, 3:1953–1962, 2015. doi: 10.1109/ACCESS.2015.2491921.
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- [10] Taban Majeed, Rasber Rashid, Dashti Ali, and Aras Asaad. Issues associated with deploying cnn transfer learning to detect covid-19 from chest x-rays. *Physical and Engineering Sciences in Medicine*, 43:1289—1303, 2020. URL <https://doi.org/10.1007/s13246-020-00934-8>.
- [11] Ruben Winastwan. Class activation mapping (cam): Better interpretability in deep learning models, 2024. URL <https://zilliz.com/learn/class-activation-mapping-CAM>.
- [12] YongWo. Imagenette - kaggle, 2024. URL <https://www.kaggle.com/datasets/yongwo/imagenette>.
- [13] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures. *arXiv preprint arXiv:1807.10108*, 2018.
- [14] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.
- [15] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.
- [16] Olga Belitskaya. Yale face database - kaggle, 2018. URL <https://www.kaggle.com/datasets/olgabelitskaya/yale-face-database/>.

- [17] Sanchari Dhar and Lior Shamir. Evaluation of the benchmark datasets for testing the efficacy of deep convolutional neural networks. *Visual Informatics*, 5, 10 2021. doi: 10.1016/j.visinf.2021.10.001.
- [18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ArXiv*, 2014. URL <https://doi.org/10.48550/arXiv.1409.1556>.
- [19] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.
- [20] Chitra Desai. Comparative analysis of optimizers in deep neural networks. *International Journal of Innovative Science and Research Technology*, 5(10):959–962, 2020.
- [21] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.
- [22] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech 256, Apr 2022. URL <https://doi.org/10.22002/D1.20087>.
- [23] Bhaskar Mondal. Cryptographic image scrambling techniques, 02 2019. URL https://www.researchgate.net/publication/331064749_Cryptographic_image_Scrambling_techniques.
- [24] Prarthana Madan Modak and Vijaykumar Pawar. A comprehensive survey on image scrambling techniques. *International Journal of Science and Research (IJSR)*, 4(12): 813–818, 2015.
- [25] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart. Hypermedia image processing reference. *England: John Wiley & Sons Ltd*, pages 209–213, 1996.
- [26] OpenCV. Opencv discrete fourier transform documentation, 2024. URL https://docs.opencv.org/4.x/d8/d01/tutorial_discrete_fourier_transform.html.

- [27] S. Agarwal, O.P. Singh, and D. Nagaria. Analysis and comparison of wavelet transforms for denoising mri image. *Biomed Pharmacol J*, 10(2), 2017. Article.
- [28] Gheyath Othman and Diyar Qader Zeebaree. The applications of discrete wavelet transform in image processing: A review. *Journal of Soft Computing and Data Mining*, 1(2): 31–43, 2020.
- [29] PyWavelet Developers. Pywavelets documentation, 2024. URL <https://pywavelets.readthedocs.io/en/latest/ref/2d-dwt-and-idwt.html>.
- [30] Suhel Dhanani and Michael Parker. 4 - introduction to digital filtering. In Suhel Dhanani and Michael Parker, editors, *Digital Video Processing for Engineers*, pages 19–28. Newnes, Oxford, 2013. ISBN 978-0-12-415760-6. doi: <https://doi.org/10.1016/B978-0-12-415760-6.00004-0>. URL <https://www.sciencedirect.com/science/article/pii/B9780124157606000040>.
- [31] OpenCV. Opencv image filtering documentation, 2024. URL https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9.
- [32] Sai Teja Erukude. Identifying bias in cnn image classification using image scrambling and transforms, 2024. URL <https://github.com/SaiTeja-Erukude/identifying-bias-in-dnn-classification>.