Impact of Image Compression on the Performance of Steel Surface Defect Classification with a CNN

Tajeddine Benbarrad 1,*  Lamiae Eloutouate 2, Mounir Arioua 1, Fatiha Elouaai 2 and My Driss Laanaoui 3

1 Laboratory of Information and Communication Technologies (LabTIC), National School of Applied Sciences of Tangier, Abdelmalek Essaadi University, Tangier 90000, Morocco; m.arioua@uae.ac.ma
2 Laboratory of Informatics Systems and Telecommunications (LIST), Faculty of Sciences and Technologies of Tangier, Abdelmalek Essaadi University, Tangier 90000, Morocco; lamiae.eloutouate@uae.ac.ma (L.E.); elouaafi@gmail.com (F.E.)
3 Laboratory of Computer and Systems Engineering (L2IS), Higher Normal School, Cadi Ayyad University of Marrakech, Marrakech 40000, Morocco; d.laanaoui@uca.ma
* Correspondence: tbenbarrad@uae.ac.ma

Abstract: Machine vision is increasingly replacing manual steel surface inspection. The automatic inspection of steel surface defects makes it possible to ensure the quality of products in the steel industry with high accuracy. However, the optimization of inspection time presents a great challenge for the integration of machine vision in high-speed production lines. In this context, compressing the collected images before transmission is essential to save bandwidth and energy, and improve the latency of vision applications. The aim of this paper was to study the impact of quality degradation resulting from image compression on the classification performance of steel surface defects with a CNN. Image compression was applied to the Northeastern University (NEU) surface-defect database with various compression ratios. Three different models were trained and tested with these images to classify surface defects using three different approaches. The obtained results showed that trained and tested models on the same compression qualities maintained approximately the same classification performance for all used compression grades. In addition, the findings clearly indicated that the classification efficiency was affected when the training and test datasets were compressed using different parameters. This impact was more obvious when there was a large difference between these compression parameters, and for models that achieved very high accuracy. Finally, it was found that compression-based data augmentation significantly increased the classification precision to perfect scores (98–100%), and thus improved the generalization of models when tested on different compression qualities. The importance of this work lies in exploiting the obtained results to successfully integrate image compression into machine vision systems, and as appropriately as possible.

Keywords: machine vision; convolutional neural network; image compression; steel surface defect classification

1. Introduction

Automation is one of the major challenges of Industry 4.0. It consists of optimizing industrial processes with automated systems and integrating technologies into manufacturing processes to increase productivity and autonomy, improve labor conditions, and simplify certain operations [1]. However, in the context of Industry 4.0, equipment automation must be combined with efficient data exchange to build production systems that enable smart, decentralized, and data-informed decision making while minimizing human interaction with processes [2].

Nowadays, industrial automation has become a concept intrinsically linked to the Internet of Things (IoT). The exploitation of generated data by IoT sensors makes it possible for machines to communicate with each other and determine actions in real time to
adapt immediately to the requirements, from manufacturing to maintenance and even to market demands. The data coming from the physical environment are then sent to a web platform and processed, thus facilitating decision making, especially in process changes [3]. Concretely, IoT enables to develop the interconnectivity of the different tools and systems of the manufacturing chain by exploiting the intelligent sensors data using big data and artificial intelligence (AI), which include numerous advantages and benefits for main industrial operations such as manufacturing process automation and monitoring [4], predictive maintenance [5], resource and inventory management [6], and quality control [7].

In addition, machine vision (MV) represents a key technology and a powerful support for IoT automation solutions [8]. MV is an AI technique that allows the analysis of images captured by cameras. It is capable of recognizing an image, understanding it, and processing the resulting information [9]. The technical advances in terms of cameras and lighting systems, as well as computer resources, have considerably expanded the field of application of MV and have opened up this technology to all industrial sectors [10]. MV is an extremely strong complement to IoT automation technologies. Automated inspection systems can work faster and more accurately than manual quality control, and they immediately surface relevant data for decision makers when defects and exceptions are detected. MV provides several advantages and performs various controls that would otherwise require different equipment [11].

The main components of an MV system are lighting, lens, image sensor, vision processing, and communications. Lighting illuminates the part to be inspected, highlighting its features so that they are clearly visible to the camera. The lens then captures the image and presents it to the sensor. Finally, the sensor converts this light into a digital image that is sent for analysis [12].

Vision processing refers to the mechanism of extracting information from the captured image; it can be performed internally in a standalone vision system or externally in a PC-based system. Advances in AI, specifically deep learning (DL) algorithms, have brought a revolution to MV by introducing nontraditional and efficient solutions that have made it possible to create high-performance vision applications. Indeed, the advent of the convolutional neural network (CNN) has opened up MV to the industrial sector, and has made this technology an attractive investment for companies seeking to automate tasks. CNNs can identify features that are not visible to humans, facilitate studies, and automate actions, thus saving a lot of time and energy [13].

MV is increasingly becoming a key component of the steel industry’s production lines [14,15]. Due to the limitations of the production conditions, the surface of metals inevitably shows various types of defects; for example, scratches, surface cracks, and rolling scale. These defects not only affect the appearance of the product, but also reduce the properties of the product such as corrosion resistance and fatigue strength, which can result in huge economic waste. Automatic surface defect inspection has become a major necessity in the metal fabrication industry, as it detects manufacturing defects with high accuracy and speed. By using a new DL-based approach, it is possible to inspect all types of difficult metal surfaces with precision and repeatability [16,17].

The images captured by cameras can be stored and processed at the edge of the network itself or on a remote server. However, due to the storage and processing capabilities of IoT objects, which are often very limited due to their size, energy, power, and computational capacity, processed images are generally sent to a remote server for analysis and storage [18]. Furthermore, the communication between IoT devices is primarily wireless, as they are usually installed in geographically dispersed locations. Wireless channels are unreliable and often present high distortion rates. Therefore, the main challenge is to ensure that the appropriate type of image is obtained, with the most-optimized size and at the right quality level. In this context, compression of collected images before transmission and storage is essential to reduce costs in terms of bandwidth and storage capacity [19].

Image compression is the application of data compression to digital images by reducing the redundancy of data in images. The purpose of compression is to reduce the memory
capacity required for image storage and to accelerate its transmission [20]. The objective of this research was to study the impact of the quality degradation resulting from image compression on the performance of steel surface defect classification with a CNN. Figure 1 presents an example of a compressed image with three different parameters, resulting in different image quality degradations. Image classification was successful for the first two qualities, but failed for the most degraded image. The image used in this illustration was taken from the Northeastern University (NEU) surface-defect database used in the performed experiments in this research [21].

Initially, we created several degraded image datasets by applying standard image compression algorithms to an image database. Afterwards, we performed three different experiments using these datasets: In the first one, we trained and tested the models with compressed image datasets with the same compression parameters. In the second experiment, we trained the models using a compressed dataset with a certain quality, but tested them using all other compressed datasets with the different qualities. In the third experiment, we evaluated the impact of training models with compression-based data augmentation on the classification performance of CNNs. Each model was trained once with a mixture of the different qualities and then evaluated on all compressed datasets.

This paper’s main contributions are as follows:

- Identifying the parameters that can be used to compress images as much as possible, without losing the accuracy of classification with a CNN;
- Evaluating the impact of image compression on the classification performance of a CNN that is trained and tested using compressed image datasets with the same parameters;
- Investigating the impact of image compression on the classification performance of a trained and tested CNN using compressed image datasets with different parameters;
- Studying the benefit of compression-based data augmentation on the classification performance of a CNN.

This paper is organized as follows. Related work is presented in Section 2. Section 3 outlines the theoretical background. Section 4 provides a detailed description of the methodology. Section 5 presents an overview of the results and discussion. Section 6 concludes the paper.

2. Related Work

The availability of massive amounts of data in images has enabled the application of DL models, and in particular CNNs, which now surpass various machine learning approaches in performance and are widely used for a variety of different tasks. However, storage and transmission of large amounts of images are challenging. Therefore, compressing the collected images before transmission is essential to save bandwidth and energy, and
improve the latency of vision applications. Considering the image degradation induced by different lossy compression algorithms affecting the performance of CNN models that are vulnerable to image manipulation, various research activities have been performed in the past few years to study the impact of image compression on the performance of CNNs.

Jo, Y.Y. et al. [22] analyzed the impact of image compression on the performance of DL-based models for classifying mammograms into “malignant” cases that lead to cancer diagnosis and treatment, or “normal” and “benign” nonmalignant cases that do not require immediate medical intervention. This paper showed that training on images using augmentation based on compression improved models when tested on compressed data, and that moderate image compression did not have a substantial impact on the classification performance of the DL-based models.

Bouderbal, I. et al. [23] analyzed some image preprocessing techniques for real-time object detection in the context of autonomous vehicles. They examined the impact of image resolution and compression on the accuracy of road object detection. To this end, several experiments were performed on the state-of-the-art YOLOv3 detector. The experimental results showed that the array was resilient to compression, given that its level was sufficient.

Poyser, M. et al. [24] investigated the impact of common image and video compression techniques on the performance of DL architectures. They focused on JPEG and H.264 (MPEG-4 AVC), which are lossy image and video compression techniques commonly used in network-connected image and video devices and infrastructures. The impact on the performance of five distinct tasks was analyzed: human pose estimation, semantic segmentation, object detection, action recognition, and monocular depth estimation. The results of this study revealed a nonlinear and nonuniform relationship between network performance and the applied level of lossy compression.

Steffens, C.R. et al. [25] evaluated the robustness of several high-level image recognition models and examined their performance in the presence of different image distortions. They proposed a testing framework that emulated bad exposure conditions, low-range image sensors, lossy compression, and commonly observed noise types. The results of this work in terms of accuracy, precision, and F1 score indicated that most CNN models were marginally affected by mild miss-exposure, heavy compression, and Poisson noise. On the other hand, severe exposure defects, impulse noise, or signal-dependent noise resulted in a substantial decrease in accuracy and precision.

Ghazvinian, Z. et al. [26] investigated the impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images. The authors found that their CNN model was robust against compression ratios up to 24:1 when it was trained on uncompressed high-quality images. They also demonstrated that a model trained on lower quality images—i.e., lossy compressed images—depicted a classification performance that was significantly improved for the corresponding compression ratio.

Manual surface inspection is a time- and effort-consuming process, which makes automation of surface defect classification very important for product quality control in the steel industry. However, the traditional methods cannot be properly applied on production lines due to their low accuracy and slow speed. Accordingly, several methods of automatic surface defect inspection have been proposed in previous research. Gradually, researchers have focused on developing new approaches based on deep neural networks for the analysis of steel surfaces in order to improve the classification accuracy and speed [27–29]. Other works have investigated the potential of transfer learning methods for the steel defect classification problem in order to overcome the DL training issues that require large processing capacity, especially when dealing with large amounts of data [30–32]. Several researchers have studied some specific types of steel surface defects, such as scratches, scrapes, abrasions, and cracks, in order to improve the detection and classification performance for these types of defects [33,34].

Our review of related works has shown that most research has focused on the study and development of DL models in order to reduce implementation costs and improve the
efficiency of surface-defect inspection systems. However, little emphasis has been placed on images, which are a key component for training and building successful DL models. The objective of this paper was to evaluate the impact of quality degradation resulting from image compression on the performance of steel surface defect classification with a CNN. The results of this research can be exploited to integrate image compression into surface-defect inspection systems in order to reduce bandwidth and storage costs, and improve latency.

3. Theoretical Background

DL is an AI approach that is derived from machine learning, in which the machine is capable of learning on its own, as opposed to programming, where it simply executes predefined rules [35]. DL is based on an artificial neural network that imitates the human brain in processing data and creating models that are used in decision making. This network consists of tens or even hundreds of neural layers, each layer receiving and interpreting information from the previous layer [36]. Incorrect answers are eliminated at each step and sent back to the upstream levels to adjust the mathematical model. Progressively, the program reorganizes the information into more complex blocks. When this model is subsequently applied to other cases, it will normally be able to solve problems that it has never encountered before. Training data is crucial for building DL models. Indeed, the system performs better when it accumulates different experiences. DL is used in many fields: image recognition [37], automatic translation [38], autonomous driving [39], intelligent robots [40], etc.

DL has often been proposed in image recognition for MV applications and has shown promising performances; it uses CNNs to perform classification tasks by identifying features from learning images [41]. CNNs are a particular form of multilayer neural network whose connection architecture is inspired by the visual cortex of mammals. Their conception is based on the discovery of visual mechanisms in living organisms, which allows them to categorize information from the most simple to the most complex. A CNN architecture consists of a succession of processing blocks to extract the features that discriminate the image class from the others. A processing block is composed of one or several: convolution layers that analyze the characteristics of the input image, correction layers, often called “ReLUs” in reference to the activation function (rectified linear units), and pooling layers that reduce the size of the intermediate image. The blocks follow each other until the final layers of the network, which classify the image and calculate the error between the prediction and the target value: the fully connected layer and the loss layer. This is the way in which the convolution, correction, and pooling layers are interconnected in the processing blocks, as well as the processing blocks between them, which determines the particularity of the network architecture. This architecture is defined as a result of applied research work [42,43].

The convolution layer is a stack of convolutions. Indeed, several convolution kernels traverse the image and generate several output feature maps. The specific parameters of each convolution kernel are defined according to the information that is sought in the image [44]. The correction or activation layer is the application of a nonlinear function to the output feature maps of the convolution layer. Making the data nonlinear facilitates the extraction of complex features that cannot be modeled by a linear combination of a regression algorithm. Rectified linear units (ReLUs) is the most widely used activation function [45]. The formula of this function is given in Equation (1):

\[ f(x) = \max(0, x) \]  

The pooling step is a subsampling process. Generally, a pooling layer is inserted regularly between the correction and convolution layers. By reducing the size of the feature maps, and thus the number of network parameters, the computation time is accelerated, and the risk of overfitting is reduced [46]. The fully connected layer classifies the image using the extracted features from the processing block sequence. It is fully connected.
because all the inputs of the layer are connected to the output neurons of the layer. Each neuron attributes to the image a probability value of belonging to a given class among the possible classes [47]. The loss layer is the final layer of the network. It calculates the error between the network prediction and the actual value. In a classification task, the random variable is discrete, as it can take only the values 0 or 1, representing the belonging (1) or not (0) to a class. This is why the most common and suitable loss function is the cross-entropy function [48]. The formula of this function is given in Equation (2):

\[
\text{loss}(x, \text{class}) = -\sum_{\text{class}=1}^{C} y_{x, \text{class}} \log (p_{x,\text{class}}) \quad (2)
\]

A CNN is simply a stack of several layers: convolution, pooling, ReLU correction, and fully connected, as shown in Figure 2. Each image received as input will be filtered, reduced, and corrected several times, to finally form a vector. In the classification problem, this vector contains the class affiliation probabilities.

![Figure 2. Common types of layers in basic CNNs.](image)

Compression is the process of reducing the number of bits that are needed to represent data. Compressing data optimizes storage capacity and file-transfer speed. It reduces costs in both areas. Compression algorithms are distinguished by three essential parameters: the compression ratio, the compression quality, and the speed of compression and decompression [49]. Compression can be done with or without loss. Lossless compression keeps the original file intact and allows the restoration of its original state without losing any bits during decompression. This method is commonly used to compress executable, text, and worksheet files in which the loss of words or numbers can modify the information. Lossy compression definitively removes redundant or unimportant bits by degrading the quality of the original file to reduce further the storage size [50]. This approach is generally applied to audio or visual data, which may be significantly altered without being perceptible to humans.

The Joint Photographic Experts Group (JPEG) format is a lossy compression method that achieves a high compression ratio with very correct quality. These two benefits make it one of the most popular image formats, particularly on the web, where storage and transfer problems are important. The algorithm is especially effective on images with smooth color variations, such as photographs. The principle of the JPEG algorithm for image compression is as follows. An image is sequentially decomposed into blocks of \(8 \times 8\) pixels, and the compression then works only on the pixel blocks [51]. Then, a discrete cosine transform (DCT) is applied to each block of pixels. The DCT operation evaluates the amplitude of changes from one pixel to another in order to identify high and low frequencies [52]. Afterwards, the quantization attenuates the high frequencies of the image that have been detected by the DCT [53]. Indeed, quantization reduces the importance of
contrast areas (high frequencies) that are not easily recognized by the human eye [54]. The main limitations of the JPEG compression algorithm is the tiling effect that appears at a high compression ratio and the destruction and irreversibility of compression.

4. Methodology

Compression reduces image size and optimizes costs in terms of bandwidth and storage memory. Therefore, image compression can improve latency in MV applications by reducing image transfer and processing time, which will improve the performance of these applications and ensure their integration into high-speed production lines. Considering the image degradation induced by the different lossy compression algorithms that can decrease the accuracy of image identification, a study of the impact of these techniques on image classification with CNNs is elaborated here in order to determine the best way to integrate image compression to MV systems without losing classification precision.

In this study, we performed three different experiments. In the first experiment, the datasets used to train and test the models were compressed with the same compression parameters. In the second experiment, we trained the models with one dataset that was compressed with a certain quality, but tested them with all other datasets that were compressed with the different qualities. In the third experiment, we studied the impact of training models with a compression-based data augmentation. Each model was trained once with a mixture of the different qualities and then evaluated on all compressed datasets. The compression parameters used in these experiments are presented in Table 1. In addition, we also noted the compression ratio \( C_R \), which evaluated the compression efficiency of an algorithm for an image [55]. We used the formula given in Equation (3) to calculate the \( C_R \):

\[
C_R = \frac{n_1}{n_2}
\]

where \( n_1 \) is the original image size and \( n_2 \) is the compressed image size.

Table 1. The different compression parameters used in the experiments.

| Designation | Scale | Quality | \( C_R \) |
|-------------|-------|---------|----------|
| Q1          | 1/1   | 5       | 27.32    |
| Q2          | 1/1   | 10      | 21.69    |
| Q3          | 1/1   | 20      | 15.46    |
| Q4          | 1/1   | 30      | 12.41    |
| Q5          | 1/1   | 40      | 10.63    |
| Q6          | 1/1   | 50      | 9.33     |
| Q7          | 1/1   | 60      | 8.21     |

Three different models were investigated in this study. The first was a simple CNN model with three convolutional layers followed by two dense layers and an output layer with six classes, as shown in Figure 3.

Figure 3. Architecture of the simple CNN model.
Three different models were investigated in this study. The first was a simple CNN model. The architecture of the simple CNN model is shown in Figure 4.

The second model was Vgg16, a very deep CNN with a very high number of parameters. Due to its depth and the number of fully connected nodes, it takes too much time to train [56]. Vgg16 has five blocks of convolutional layers, in which we used rectified linear units (ReLUs) as the activation function and MaxPooling for downsampling in between each convolutional block. After the last convolution and MaxPooling layer, we passed the data to the dense layer. For this, we flattened the vector that came out of the convolutions and added two dense layers of 4096 units and a dense Softmax layer of 6 units. The Vgg16 architecture used in this experiment is presented in Figure 4.

The third model was MobileNet, a vision model for TensorFlow designed to efficiently maximize accuracy while taking into account the limited resources of an embedded or device-based application. MobileNet is small, low-latency, low-power model that is configured to meet the resource constraints of a variety of use cases. MobileNet’s architecture is built on depthwise separable convolution layers, except for the first layer, which is a full convolutional layer. Each depthwise separable convolution layer consists of a depthwise convolution and a pointwise convolution. Counting depthwise and pointwise convolutions as separate layers, a MobileNet has 28 layers [57]. MobileNet’s architecture used in this experiment is displayed in Figure 5.

We used a dense layer of 6 units at the end of all models with a Softmax activation, as we had 6 classes to predict in the output, which were the six surface-defect types contained in the Northeastern University (NEU) surface-defect database. Finally, it is important to mention that the three models with distinct architectures were chosen in order to study the impact of image compression on different types of CNN models with different characteristics.

For data collection, the Northeastern University (NEU) surface-defect database was used to train and test our models [29]. The dataset, which contains 1800 (200 × 200) gray scale images, with 300 samples each of six different kinds of typical surface defects, is published by Northeastern University. Each image identifies a typical kind of surface defect of the hot-rolled steel strip that occurs on steel surfaces during the fabrication process. Our aim was to make the models classify those defects into six categories: crazing (Cr), inclusion (In), patches (Pa), pitted surface (PS), rolled-in scale (RS), and scratches (Sc). The NEU surface-defect database presents two challenges: intraclass defects have large differences in appearance, while interclass defects have similar appearances, and defect
images are influenced by lighting and material changes. Sample images of six types of surface defects are shown in Figure 6.

![Sample images of six types of surface defects](image_url)

**Figure 6.** Samples of six kinds of typical surface defects from the NEU surface-defect database. Each row shows one example image from each of 300 samples of a class [21,29]. Figure 6 is Adapted from [21], with permission from © 2013 Elsevier.

For validation, the dataset was divided at a ratio of 80:20: 80% of the dataset was used as training data, and the remaining 20% was used to test the models after training. The developed models were trained for 20 epochs with the Adam optimizer [58] using a batch size of 30. The original dataset may not have been sufficient to train a deep CNN. Therefore, we used the ImageDataGenerator class of the Tensorflow API to generate an additional set for training. The surface-defect classification was written in Python. The models were developed using the KERAS package and executed on a Google Colab high-performance GPU.

Evaluating a DL model is as important as creating it. We created models to run on new and unseen data. Therefore, extensive evaluation was necessary to create a robust model. In this experiment, we dealt with a classification problem: we used labeled data to predict to which class an object belonged. Therefore, we used a confusion matrix to evaluate our models. The confusion matrix was a cross-table between the actual values and the predictions that went beyond classification accuracy by showing the correct and incorrect predictions for each class, as shown in Figure 7. This matrix identified four categories of outcomes:

- True Positive (TP): predicted a positive class as positive;
- False Positive (FP): predicted a negative class as positive;
- False Negative (FN): predicted a positive class as negative;
- True Negative (TN): predicted a negative class as negative.
Evaluating a DL model is as important as creating it. We created models to run on new and unseen data. Therefore, extensive evaluation was necessary to create a robust model. In this experiment, we dealt with a classification problem: we used labeled data to predict to which class an object belonged. Therefore, we used a confusion matrix to evaluate our models. The confusion matrix was a cross-table between the actual values and the predictions that went beyond classification accuracy by showing the correct and incorrect predictions for each class, as shown in Figure 7. This matrix identified four categories of outcomes:

- **True Positive (TP):** predicted a positive class as positive;
- **False Positive (FP):** predicted a negative class as positive;
- **False Negative (FN):** predicted a positive class as negative;
- **True Negative (TN):** predicted a negative class as negative.

![Confusion matrix for binary classification.](image)

Several performance indicators could be derived from the confusion matrix. Recall measured the capacity of our model to correctly predict positive classes. The focus of recall was the TP classes. It indicated the number of positive classes that the model correctly predicted. We used the formula given in Equation (4) to calculate the recall. Precision indicated the quality of our model when the prediction was positive. Precision examined the positive predictions, and indicated how many positive predictions were true. We used the formula given in Equation (5) to calculate the precision. The F1 score was the weighted average of precision and recall, and included both FP and FN. The F1 score was defined by the formula given in Equation (6):

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (4) \\
\text{Precision} = \frac{TP}{TP + FP} \quad (5) \\
\text{F1}_\text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \quad (6)
\]

In addition, we were particularly interested in the behavior of two major metrics used to evaluate compression encoders with respect to the obtained performance of models: The peak signal-to-noise ratio (PSNR) and the structural similarity (SSIM).

PSNR is a measure of distortion used in digital images, particularly in image compression. It is the most commonly used measure to quantify the performance of encoders by evaluating the difference between the original and compressed representations at a pixel-by-pixel level [59]. PSNR is defined by the formula given in Equation (7):

\[
\text{PSNR} = 10 \times \log_{10} \left( \frac{\text{MAX}_I^2}{\text{MSE}} \right) = 20 \times \log_{10} \left( \frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right) \quad (7)
\]

where \(\text{MAX}_I\) represents the maximum pixel value in the original image, and MSE is the mean-squared error defined by the formula given in Equation (8):

\[
\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \quad (8)
\]

where \(Y_i\) represents the pixels of the original image, \(\hat{Y}_i\) represents the pixels of the compressed image, and \(n\) is the number of pixels of the image.
PSNR assesses how close the compressed image is to the original in terms of signal. However, it does not reflect the visual quality of reconstruction, and cannot be considered as an objective metric of the visual quality of an image. Therefore, SSIM was developed to evaluate the visual quality of a compressed image compared to the original image [60]. Unlike PSNR, SSIM is based on the visible structures in the image. Its purpose is to measure the similarity between two given images, instead of the pixel-to-pixel difference as is done by PSNR. SSIM measured between a compressed image \( x \) and an original image \( y \) is defined by the formula given in Equation (9):

\[
SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\text{cov}_{xy} + c_2)}{\mu_x^2 + \mu_y^2 + c_1}(\sigma_x^2 + \sigma_y^2 + c_2)
\]

where:
- \( \mu_x \) is the average of \( x \) and \( \mu_y \) is the average of \( y \);
- \( \sigma_x^2 \) is the variance of \( x \) and \( \sigma_y^2 \) is the variance of \( y \);
- \( \text{cov}_{xy} \) is the covariance of \( x \) and \( y \);
- \( c_1 = (k_1 L)^2 \) and \( c_2 = (k_2 L)^2 \) are two variables to stabilize the division with a weak denominator;
- \( L \) is the dynamic range of the pixel values (typically, this is 2\(^{2L} \text{bits per pixel} - 1 \));
- \( k_1 = 0.01 \) and \( k_2 = 0.03 \) by default.

5. Results and Discussion

Once training was complete in the first experiment, we calculated the performance of our trained models using the compressed test datasets with the same compression parameters of training images. The obtained results are reported in Table 2. All results in this section are presented under the format (precision, recall, F1 score). The results showed that all models maintained approximately the same precision and recall for all used compression qualities. In fact, the CNN compared the images fragment by fragment in order to identify approximate features that were similar. By finding similar features that contained the most common aspects of the images, the CNN successfully identified the appropriate class for each image. This explains why image compression did not impact the classification performance of CNN when the training and test images were compressed with the same parameters, since this compression resulted in images with a high degree of similarity.

Table 2. Classification performance of models when trained and tested on images with the same compression qualities.

| Model/Data | Q1     | Q2     | Q3     | Q4     | Q5     | Q6     | Q7     |
|------------|--------|--------|--------|--------|--------|--------|--------|
| CNN3       | (0.85, 0.83, 0.82) | (0.89, 0.87, 0.87) | (0.86, 0.86, 0.86) | (0.84, 0.81, 0.80) | (0.89, 0.88, 0.87) | (0.83, 0.81, 0.81) | (0.86, 0.82, 0.82) |
| MobileNet  | (0.97, 0.97, 0.97) | (0.99, 0.99, 0.99) | (0.98, 0.98, 0.98) | (0.98, 0.98, 0.98) | (0.98, 0.98, 0.98) | (0.98, 0.98, 0.98) | (1.00, 1.00, 1.00) |
| Vgg16      | (0.92, 0.91, 0.91) | (0.92, 0.91, 0.91) | (0.94, 0.93, 0.92) | (0.94, 0.93, 0.93) | (0.90, 0.90, 0.90) | (0.93, 0.92, 0.91) | (0.92, 0.90, 0.90) |

The classification performance of models that were trained and tested on images with different compression qualities in the second experiment are displayed in Table 3. For each table, the ”M-Q” rows show the classification performance for a trained model on images compressed with Q quality, and tested on data compressed with the quality indicated in the column header. The obtained results clearly showed that the classification efficiency was affected when the training and test datasets were compressed using different parameters. This impact was more obvious when there was a large difference between these compression parameters. In addition, when we compared the performance of the three investigated models, we saw that the performance deterioration was much more significant for models that reached a very high accuracy during training. Indeed, very efficient models extracted similar features from images with high precision. This explains why MobileNet and Vgg16 were more affected than our simple three-layer CNN. The
In order to analyze the behavior of the metrics used to evaluate the compression encoders in relation to the obtained performance of models in this experiment, we calculated the PSNR and SSIM between the different compression qualities. The reported results in the Table 4, under the format (PSNR/SSIM), show that the classification accuracy degraded as the SSIM values decreased, indicating an obvious difference in visual quality between the training and test images. These results demonstrated that the performed classification by CNN was related to the structure evaluation of images. However, the performed measurements did not present a clear correlation between PSNR values and classification performance. Therefore, SSIM was more appropriate for evaluating the classification performance of CNNs with respect to image degradation induced by lossy compression. This makes a lot of sense, since SSIM was introduced to mimic the subjective assessment of image quality by human vision systems, and CNNs are based on a connection architecture inspired by the visual cortex of mammals. The comparison between the SSIM graph curves, as shown in Figure 9, and precision graph curves of trained and tested models on different compression qualities, as shown in Figure 8, clearly show this correlation. The curves with the same color followed approximately the same behavior.

Table 3. Classification performance of models when trained and tested on images with different compression qualities.

| Model/Data | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 |
|------------|----|----|----|----|----|----|----|
| CNN3       | 0.85, 0.83, 0.82 | 0.84, 0.81, 0.81 | 0.78, 0.74, 0.72 | 0.75, 0.71, 0.69 | 0.73, 0.71, 0.68 | 0.70, 0.69, 0.66 | 0.69, 0.69, 0.66 |
| MobileNet  | 0.81, 0.67, 0.63 | 0.90, 0.89, 0.88 | 0.88, 0.86, 0.86 | 0.85, 0.83, 0.82 | 0.84, 0.81, 0.81 | 0.82, 0.78, 0.79 | 0.82, 0.78, 0.79 |
| Vgg16      | 0.78, 0.73, 0.68 | 0.84, 0.79, 0.78 | 0.85, 0.81, 0.81 | 0.84, 0.81, 0.80 | 0.83, 0.79, 0.79 | 0.84, 0.78, 0.78 | 0.81, 0.76, 0.76 |
|            | 0.76, 0.67, 0.66 | 0.92, 0.91, 0.91 | 0.91, 0.90, 0.90 | 0.90, 0.88, 0.88 | 0.89, 0.88, 0.87 | 0.89, 0.88, 0.87 | 0.88, 0.86, 0.86 |
|            | 0.83, 0.81, 0.80 | 0.87, 0.86, 0.85 | 0.86, 0.84, 0.84 | 0.84, 0.83, 0.83 | 0.85, 0.83, 0.83 | 0.83, 0.81, 0.81 | 0.83, 0.81, 0.81 |
|            | 0.83, 0.81, 0.80 | 0.90, 0.87, 0.87 | 0.87, 0.85, 0.85 | 0.86, 0.83, 0.83 | 0.87, 0.84, 0.84 | 0.86, 0.83, 0.83 | 0.86, 0.82, 0.82 |

Table 4. PSNR and SSIM measurements between different compression qualities.

| Compression Quality | PSNR/Q1 | PSNR/Q2 | PSNR/Q3 | PSNR/Q4 | PSNR/Q5 | PSNR/Q6 | PSNR/Q7 |
|---------------------|---------|---------|---------|---------|---------|---------|---------|
| Q1                  | 27.88/0.7073 | 28.23/0.6950 | 28.27/0.6887 | 28.23/0.6830 | 28.19/0.6784 | 28.16/0.6742 |
| Q2                  | 31.04/0.7876 | 34.49/0.8848 | 33.72/0.8593 | 33.91/0.8611 | 34.12/0.8666 | 34.12/0.8666 |
| Q3                  | 31.24/0.8081 | 34.49/0.8848 | 33.72/0.8593 | 33.91/0.8611 | 34.12/0.8666 | 34.12/0.8666 |
| Q4                  | 31.24/0.8081 | 34.49/0.8848 | 33.72/0.8593 | 33.91/0.8611 | 34.12/0.8666 | 34.12/0.8666 |
| Q5                  | 33.72/0.8593 | 36.59/0.9260 | 35.53/0.9007 | 35.53/0.9007 | 35.51/0.8864 | 35.51/0.8864 |
| Q6                  | 33.91/0.8611 | 35.53/0.9007 | 35.53/0.9007 | 35.53/0.9007 | 35.51/0.8864 | 35.51/0.8864 |
| Q7                  | 34.12/0.8666 | 35.53/0.9007 | 35.53/0.9007 | 35.53/0.9007 | 35.51/0.8864 | 35.51/0.8864 |
Figure 8. Precision graph of models that were trained and tested on images with different compression qualities: (a) CNN3; (b) Vgg16; (c) MobileNet.

Figure 9. SSIM measurements between different compression qualities.
Given that models are trained and evaluated on a wider variety of image types, and data storage guidelines regarding image compression vary across the various use cases, we investigated the impact of training classifiers using data that was compressed with a mixture of compression qualities. Instead of using the ImageDataGenerator class of the Tensorflow API to generate an additional training set by applying many geometric transformations for data augmentation (rescale, rotation, flip, zoom, etc.), we increased the training dataset by adding all the compressed images with all the compression qualities used in this work. Consequently, we trained our models on a dataset containing \((1440 \times 7 = 10,080)\) images, with 1440 images for each compression quality. Table 5 summarizes the classification performance of trained models with compression-based data augmentation. The obtained results proved that compression-based data augmentation dramatically increased the classification efficiency of our models, even when evaluated with the different compression qualities. This indicated that using compression for data augmentation improved the generalization of models when tested on different compression qualities. This can be explained by the fact that CNNs successfully identify the appropriate class for each image by finding similar features that contain the most common aspects of images. When we increased the training dataset using compression, we actually multiplied the number of training images by 7. This improved the feature extraction by including features from the compressed images with the different parameters used in the experiment. Afterwards, when a compressed image with a certain quality was introduced, the network identified similar features and successfully classified the image. Figure 10 shows the training accuracy, validation accuracy, precision, and recall curves per epoch for the trained models with compression-based data augmentation.

| Model/Data Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Q7 |
|---------------|----|----|----|----|----|----|
| CNN3          | (0.92, 0.91, 0.91) | (0.98, 0.98, 0.98) | (0.99, 0.99, 0.99) | (0.99, 0.99, 0.99) | (0.99, 0.99, 0.99) | (0.99, 0.99, 0.99) |
| MobileNet     | (0.94, 0.92, 0.92) | (0.98, 0.97, 0.98) | (0.99, 0.99, 0.99) | (1.00, 1.00, 1.00) | (0.99, 0.99, 0.99) | (0.99, 0.99, 0.99) |
| Vgg16         | (0.96, 0.96, 0.96) | (0.99, 0.99, 0.99) | (1.00, 1.00, 1.00) | (1.00, 1.00, 1.00) | (1.00, 1.00, 1.00) | (1.00, 1.00, 1.00) |

Figure 10. Left: training and validation accuracy curves; right: Precision and recall curves, per epoch for the trained models with compression-based data augmentation: (a) CNN3; (b) Vgg16; (c) MobileNet.
Table 5. Classification performance of trained models with compression-based data augmentation.

| Model/Data | Q1       | Q2       | Q3       | Q4       | Q5       | Q6       | Q7       |
|------------|----------|----------|----------|----------|----------|----------|----------|
| CNN3       | (0.92, 0.91) | (0.98, 0.98) | (0.99, 0.99) | (0.99, 0.99) | (0.99, 0.99) | (0.99, 0.99) | (0.99, 0.99) |
| MobileNet  | (0.94, 0.92) | (0.98, 0.97) | (0.99, 0.99) | (1.00, 1.00) | (0.99, 0.99) | (0.99, 0.99) | (1.00, 1.00) |
| Vgg16      | (0.96, 0.96) | (0.99, 0.99) | (1.00, 1.00) | (1.00, 1.00) | (1.00, 1.00) | (1.00, 1.00) | (1.00, 1.00) |

6. Conclusions

This paper evaluated the impact of quality degradation resulting from image compression on the classification performance of steel surface defects with a CNN. The obtained results showed that models trained and tested on compressed images with the same parameters maintained approximately the same classification performance for all used compression grades. Furthermore, the outcomes indicated that the classification efficiency was affected when the training and test datasets were compressed using different parameters. This impact was more evident when there was a large difference between these compression parameters, and for models that achieved very high accuracy. In addition, the findings revealed that compression-based data augmentation dramatically increased the classification performance, and thus improved the generalization of models when tested on different compression qualities. The experiments also demonstrated that the classification performance of models was correlated with the image quality, as evaluated by the SSIM metric. The importance of this work lies in the exploitation of the obtained results to successfully integrate image compression into machine vision systems, and as appropriately as possible. Using only one compression method (JPEG) in the experiments was the major limitation of this work. In the future, we would like to extend our research by studying the impact of other compression encoders on the classification performance of CNNs.

Author Contributions: Conceptualization, T.B., L.E. and M.A.; methodology, T.B. and M.A.; software, T.B. and L.E.; validation, M.A., F.E. and M.D.L.; formal analysis, M.A. and M.D.L.; investigation, T.B. and L.E.; resources, T.B.; data curation, T.B.; writing—original draft preparation, T.B. and M.A.; writing—review and editing, T.B., M.A. and F.E.; visualization, L.E.; supervision, M.A. and M.D.L.; project administration, M.A.; funding acquisition, M.D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available dataset was analyzed in this study. This data can be found here: [http://faculty.neu.edu.cn/songkc/en/zhym/263264/list/index.htm], accessed on 8 August 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghobakhloo, M.; Fathi, M.; Iranmanesh, M.; Maroufkhani, P.; Morales, M.E. Industry 4.0 ten years on: A bibliometric and systematic review of concepts, sustainability value drivers, and success determinants. J. Clean. Prod. 2021, 302, 127052. [CrossRef]
2. Bousdekis, A.; Lepenioti, K.; Apostolou, D.; Mentzas, G. A Review of Data-Driven Decision-Making Methods for Industry 4.0 Maintenance Applications. Electronics 2021, 10, 828. [CrossRef]
3. Elsisi, M.; Mahmoud, K.; Lehtonen, M.; Darwish, M.M.F. Reliable Industry 4.0 Based on Machine Learning and IoT for Analyzing, Monitoring, and Securing Smart Meters. Sensors 2021, 21, 487. [CrossRef]
4. Khairnar, V.; Kolhe, L.; Bhagat, S.; Sahu, R.; Kumar, A.; Shaikh, S. Industrial Automation of Process for Transformer Monitoring System Using IoT Analytics. In Inventive Communication and Computational Technologies; Springer: Singapore, 2020; pp. 1191–1200. [CrossRef]
5. Theissler, A.; Pérez-Velázquez, J.; Kettelgerdes, M.; Elger, G. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. Reliab. Eng. Syst. Saf. 2021, 215, 107864. [CrossRef]
6. Devasthali, A.S.; Chaudhari, A.J.; Bhutada, S.S.; Doshi, S.R.; Suryawanshi, V.P. IoT Based Inventory Management System with Recipe Recommendation Using Collaborative Filtering. In Evolutionary Computing and Mobile Sustainable Networks; Springer: Singapore, 2020, pp. 543–550. [CrossRef]

7. Shahbazi, Z.; Byun, Y.-C. Integration of Blockchain, IoT and Machine Learning for Multistage Quality Control and Enhancing Security in Smart Manufacturing. Sensors 2021, 21, 1467. [CrossRef]

8. Silva, R.L.; Junior, O.C.; Rudek, M. A road map for planning-deploying machine vision artifact in the context of industry 4.0. J. Ind. Prod. Eng. 2021, 1–14. [CrossRef]

9. Band, T.; Jie, B.Y.W.; Farid, A.A.; Lim, C.S. Machine Vision and Convolutional Neural Networks for Tool Wear Identification and Classification. In Recent Trends in Mechatronics Towards Industry 4.0; Springer: Singapore, 2021, pp. 737–747. [CrossRef]

10. Pundir, M.; Sandhu, J.K. A Systematic Review of Quality of Service in Wireless Sensor Networks using Machine Learning: Recent Trend and Future Vision. J. Netw. Comput. Appl. 2021, 188, 103804. [CrossRef]

11. Benbarrad, T.; Salhaoui, M.; Kenitar, S.; Arioua, M. Intelligent Machine Vision Model for Defective Product Inspection Based on Machine Learning. J. Sens. Actuator Netw. 2021, 10, 7. [CrossRef]

12. Luo, Y.; Li, S.; Li, D. Intelligent Perception System of Robot Visual Servo for Complex Industrial Environment. Sensors 2020, 20, 7121. [CrossRef] [PubMed]

13. Mou, H.R.; Lu, R.; An, J. Research on Machine Vision Technology of High Speed Robot Sorting System based on Deep Learning. J. Physics Conf. Ser. 2021, 1748, 022029. [CrossRef]

14. Luo, Q.; Fang, X.; Su, J.; Zhou, J.; Zhou, B.; Yang, C.; Liu, L.; Gui, W.; Tian, L. Automated Visual Defect Classification for Flat Steel Surface: A Survey. IEEE Trans. Instrum. Meas. 2020, 69, 9329–9349. [CrossRef]

15. Chu, M.-X.; Feng, Y.; Yang, Y.-H.; Deng, X. Multi-class classification method for steel surface defects with feature noise. J. Iron Steel Res. Int. 2020, 28, 303–315. [CrossRef]

16. Wang, S.; Xia, X.; Ye, L.; Yang, B. Automatic Detection and Classification of Steel Surface Defect Using Deep Convolutional Neural Networks. Metals 2021, 11, 388. [CrossRef]

17. Konovalenko, I.; Maruschak, P.; Brezinová, J.; Viňaší, J.; Brezina, J. Steel Surface Defect Classification Using Deep Residual Neural Network. Metals 2020, 10, 846. [CrossRef]

18. Benbarrad, T.; Salhaoui, M.; Arioua, M. On the Performance of Deep Learning in the Full Edge and the Full Cloud Architectures. In Proceedings of the 4th International Conference on Networking, Information Systems & Security, New York, NY, USA, 1–4 April 2021. [CrossRef]

19. Bouakazz, F.; Ali, W.; Derdour, M. Forest Fire Detection Using Wireless Multimedia Sensor Networks and Image Compression. Instrum. Mes. Métrologie 2021, 20, 57–63. [CrossRef]

20. Hussain, A.J.; Al-Fayadh, A.; Radi, N. Image compression techniques: A survey in lossless and lossy algorithms. Neurocomputing 2018, 300, 44–69. [CrossRef]

21. Song, K.; Yan, Y. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. Appl. Surf. Sci. 2013, 285, 858–864. [CrossRef]

22. Jo, Y-Y.; Choi, Y.S.; Park, H.W.; Lee, J.H.; Jung, H.; Kim, H.-E.; Ko, K.; Lee, C.W.; Cha, H.S.; Hwangbo, Y. Impact of image compression on deep learning-based mammogram classification. Sci. Rep. 2021, 11, 7924. [CrossRef]

23. Bouderbal, I.; Amamra, A.; Benatia, M.A. How Would Image Down-Sampling and Compression Impact Object Detection in the Context of Self-driving Vehicles. In Advances in Computing Systems and Applications; Springer: Cham, Switzerland, 2021; pp. 25–37. [CrossRef]

24. Poyser, M.; Atapour-Abarghouei, A.; Breckon, T.P. On the Impact of Lossy Image and Video Compression on the Performance of Deep Convolutional Neural Network Architectures. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 2830–2837. [CrossRef]

25. Steffens, C.R.; Messias, L.R.V.; Drews, P., Jr.; Botelho, S.S.D.C. Can Exposure, Noise and Compression Affect Image Recognition? An Assessment of the Impacts on State-of-the-Art ConvNets. In Proceedings of the 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), Rio Grande, Brazil, 23–25 October 2019; pp. 61–66. [CrossRef]

26. Zanjani, F.G.; Zinger, S.; Piepers, B.; Mahmoudpour, S.; Schelkens, P. Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images. J. Med. Imaging 2019, 6, 027501. [CrossRef]

27. Aydin, I.; Akin, E.; Karakose, M. Defect classification based on deep features for railway tracks in sustainable transportation. Appl. Soft Comput. 2021, 111, 107706. [CrossRef]

28. Chen, K.; Zeng, Z.; Yang, J. A deep region-based pyramid neural network for automatic detection and multi-classification of various surface defects of aluminum alloys. J. Build. Eng. 2021, 43, 102523. [CrossRef]

29. He, Y.; Song, K.; Meng, Q.; Yan, Y. An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features. IEEE Trans. Instrum. Meas. 2019, 69, 1493–1504. [CrossRef]

30. Abu, M.; Amir, A.; Lean, Y.H.; Zahri, N.A.H.; Azemi, S.A. The Performance Analysis of Transfer Learning for Steel Defect Detection by Using Deep Learning. J. Phys. Conf. Ser. 2021, 1755. [CrossRef]

31. Zhang, J.; Li, Z.; Hao, R.; Wang, X.; Du, X.; Yan, B.; Ni, G.; Liu, J.; Liu, L.; Liu, Y. Classification of Microscopic Laser Engraving Surface Defect Images Based on Transfer Learning Method. Electronics 2021, 10, 1993. [CrossRef]
32. Wan, X.; Zhang, X.; Liu, L. An Improved VGG19 Transfer Learning Experiment for Defect Recognition of Strip Steel Surface Defect Recognition Deep Neural Network Based on Few Samples and Imbalanced Datasets. *Appl. Sci.* 2021, 11, 2606. [CrossRef]

33. Konovalenko, I.; Maruschk, P.; Breusch, V.; Prentkovske, O. Recognition of Scratches and Abrasions on Metal Surfaces Using a Classifier Based on a Convolutional Neural Network. *Metals* 2021, 11, 549. [CrossRef]

34. Fu, P.; Hu, B.; Lan, X.; Yu, J.; Ye, J. Simulation and quantitative study of cracks in 304 stainless steel under natural magnetization field. *NDT E Int.* 2021, 119, 102419. [CrossRef]

35. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaria, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* 2021, 8, 53. [CrossRef]

36. Yeganeh, A.; Pourpanah, F.; Shadman, A. An ANN-based ensemble model for change point estimation in control charts. *Appl. Soft Comput.* 2021, 110, 107604. [CrossRef]

37. Chai, Q. Research on the Application of Computer CNN in Image Recognition. *J. Phys. Conf. Ser.* 2021, 1915, 032041. [CrossRef]

38. Ban, H.; Ning, J. Design of English Automatic Translation System Based on Machine Intelligent Translation and Secure Internet of Things. *Mob. Inf. Syst.* 2021, 2021, 8670739. [CrossRef]

39. Muhammad, K.; Ullah, A.; Lloret, J.; Del Ser, J.; de Albuquerque, V.H.C. Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions. *IEEE Trans. Intell. Transp. Syst.* 2020, 22, 4316–4336. [CrossRef]

40. Meng, L.; Yue Song, W.; Jin Qi, L. Design of an Intelligent Service Robot based on Deep Learning. In Proceedings of the ICIT 2020: 2020 The 8th International Conference on Information Technology: IoT and Smart City, Xi’an, China, 25–27 December 2020. [CrossRef]

41. Yu, T.; Jin, H.; Tan, W.-T.; Nahrstedt, K. SKEPRID. *ACM Trans. Multimed. Comput. Commun. Appl.* 2018, 14, 1–24. [CrossRef]

42. Yang, Q.; Li, C.; Dai, W.; Zou, J.; Qi, G.-J.; Xiong, H. Rotation Equivariant Graph Convolutional Network for Spherical Image Classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 4302–4311. [CrossRef]

43. Ling, J.; Xue, H.; Song, L.; Xie, R.; Gu, X. Region-aware Adaptive Instance Normalization for Image Harmonization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 9357–9366. [CrossRef]

44. Pavlova, M.T. A Comparison of the Accuracies of a Convolution Neural Network Built on Different Types of Convolution Layers. In Proceedings of the 2021 56th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Sozopol, Bulgaria, 16–18 June 2021; pp. 81–84. [CrossRef]

45. El Jaafari, I.; Ellahyani, A.; Charfi, S. Rectified non-linear unit for convolution neural network. *J. Phys. Conf. Ser.* 2021, 1743, 012014. [CrossRef]

46. Wu, L.; Perin, G. On the Importance of Pooling Layer Tuning for Profiling Side-Channel Analysis. In *Applied Cryptography and Network Security Workshops*; Springer: Cham, Switzerland, 2021; pp. 114–132. [CrossRef]

47. Matsumura, N.; Ito, Y.; Nakano, K.; Kasagi, A.; Tabaru, T. A novel structured sparse fully connected layer in convolutional neural networks. *Concurr. Comput. Pract. Exp.* 2021, e6213. [CrossRef]

48. Semenov, A.; Boginski, V.; Pasiliao, E.L. Neural Networks with Multidimensional Cross-Entropy Loss Functions. In *Computational Data and Social Networks*; Springer: Cham, Switzerland, 2019; pp. 57–62. [CrossRef]

49. Jayasankar, U.; Thirumal, V.; Ponnurangam, D. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *J. King Saud Univ. Comput. Inf. Sci.* 2021, 33, 119–140. [CrossRef]

50. Qasim, A.J.; Din, R.; Alyousuf, F.Q.A. Review on techniques and file formats of image compression. *Bull. Electr. Eng. Inform.* 2020, 9, 602–610. [CrossRef]

51. Isqbal, Y.; Kwon, O.-J. Improved JPEG Coding by Filtering 8 × 8 DCT Blocks. *J. Imaging* 2021, 7, 117. [CrossRef]

52. Xiao, W.; Wan, N.; Hong, A.; Chen, X. A Fast JPEG Image Compression Algorithm Based on DCT. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; pp. 106–110. [CrossRef]

53. Araujo, L.C.; Sansao, J.P.H.; Junior, M.C.S. Effects of Color Quantization on JPEG Compression. *Int. J. Image Graph.* 2020, 20, 2050026. [CrossRef]

54. Bharadwaj, N.A.; Rao, C.S.; Rahul; Gururaj, C. Optimized Data Compression through Effective Analysis of JPEG Standard. In Proceedings of the 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 5–7 March 2021; pp. 110–115. [CrossRef]

55. Ghaffari, A. Image compression-encryption method based on two-dimensional sparse recovery and chaotic system. *Sci. Rep.* 2021, 11, 1–19. [CrossRef]

56. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* 2014, arXiv:1409.1556. Available online: https://arxiv.org/abs/1409.1556 (accessed on 6 August 2021).

57. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* 2017, arXiv:1704.04861. Available online: http://arxiv.org/abs/1704.04861 (accessed on 10 August 2021).

58. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* 2014, arXiv:1412.6980. Available online: https://arxiv.org/abs/1412.6980 (accessed on 8 August 2021).
59. Choi, H.R.; Kang, S.-H.; Lee, S.; Han, D.-K.; Lee, Y. Comparison of image performance for three compression methods based on digital X-ray system: Phantom study. Optik 2018, 157, 197–202. [CrossRef]

60. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Trans. Image Process. 2004, 13, 600–612. [CrossRef]