Journal of Theoretical and Applied Information Technology

30th September 2023. Vol.101. No 18

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

STEGANALYSIS OF DATA HIDDEN IN RGB IMAGES USING RESIDUAL NEURAL NETWORK

ALANOUD MAHMOUD ALMASAR [1] , OSAMA MAHMOUD OUDA[2]

[1] Department of Computer Science, College of Computer and Information Sciences, Jouf University, Saudia Arabia

[2] Department of Computer Science, College of Computer and Information Sciences, Jouf University, Saudia Arabia

E-mail: [1]431205917@ju.edu.sa, [2]omalsayed@ju.edu.sa

ABSTRACT

Digital forensic analysis aims to apply scientific and statistical techniques to identify, gather, preserve and present relevant digital evidence which will allow to confirm or reject a hypothesis against possible criminal activity. Current methods of forensic digital analysis are effective for visual analysis of physical evidence, but they do not allow for the automatic execution of large amounts of data, for correlation studies of the files obtained, for the validation of the metadata and for the identification of abnormalities in text, graphic or audiovisuals files. For this reason, artificial intelligence techniques were proposed for processing data, for identifying patterns and trends that make it possible to perceive aspects that cannot be visually perceived. In this paper, we propose a network analysis approach to steganalysis of RGB images in frequency domain based on Residual Neural Network-50 (ResNet-50). We used the Discrete Cosine Transform (DCT) features based on standard convolutional operations in ResNet-50 generated by the first convolutional layer in the network. The DCT Feature based ResNet-50 model extensively reduces the quantity of parameters and multiplications while keeping comparable accuracy results to normal ResNet in ImageNet-1K.

Keywords: *Steganalysis, Deep Learning, Convolutional Neural Networks (CNN), Forensics.*

1. INTRODUCTION

The exchange of files over various communication channels (especially the Internet and social networks) is increasing. This file exchange can be used for legitimate personal and business purposes as well as for illegal activities. The steady growth of steganography for concealing messages in multimedia files has prompted researchers to focus on steganography and steganalysis techniques and related fields in multimedia communications. The September 11 attacks in the United States drew attention to the need for network analysis techniques to detect malicious communications from terrorists and criminals [1]. In many contexts, steganography is used to hide various types of information, such as medical, business, and personal documents, to preserve privacy. However, steganography is also used for illicit purposes, obfuscating documents exchanged in illicit transactions, such as money laundering, drug trafficking, human trafficking, and terrorism. The misuse of steganography by insiders to share confidential company documents with competitors is a very serious problem for companies.

Steganalysis is a set of techniques for distinguishing cover images from stego images (images used to carry embedded messages) [1]. The increasing malicious attacks on private, commercial and government documents by various adversaries have prompted researchers and developers in the field of information security to look for technical solutions to protect the privacy of documents sent over communication channels. This work is motivated by the rapid increase in the use of information hiding for illegal purposes. Likewise, the steganography tools for information hiding have become widely available on the internet, which made it easy for anyone to embed hidden data within a cover image. The security industry has focused on finding tools that can detect hidden messages within cover media. However, more work is needed to enhance the detection performance.

The problem we will be addressed is to deal with the widespread misuse of steganography tools for hiding secret message for illegal purposes. This work aims at presenting an efficient steganalysis technique to discover hidden data in RGB cover images. In this paper, we propose

utilizing residual Neural Network-50 (ResNet-50) to detect the hidden messages within the Discrete Cosine Transform (DCT) coefficients of RGB cover images in frequency domain. We propose and test our idea to change the input representation to train the existing ResNet-50. We use the DCT features based on standard convolutional operations in ResNet-50 generated by the first convolutional layer in the network to enhance the detection performance. The proposed model shows that ResNet can also be trained using JPEG-compressed DCT coefficients and subsequently perform better compared to traditional ResNet methods. The research is expected to improve the detection capabilities of steganalysis tools to reveal the existence of secret messages hidden in cover images. By detecting whether data is hidden in images, monitors (administrators) can further analyze suspicious images to prevent hidden data from being sent to recipients over the network.

The remainder of this paper is organized as follows. Section 2 gives brief descriptions of background concepts including the compression using the DCT and the convolutional neural network(CNN). Section 3 discusses previous related works. Section 4 introduces the proposed methodology. Section 5 discusses the results of the experimentations and Section 6 concludes the paper.

2. BACKGROUND

In this section, necessary background details required to fully understand the concepts of the proposed methods are presented.

2.1 Image Compression Using DCT

Nowadays the vast majority of digital data is visual information and consists of compressed data. Their compression is implemented with the aim of significantly reducing the size of digital files containing visuals information, while avoiding the creation of significant alterations[2]. This n reduction is significant as not only is the digital storage space reduced images and videos, but the involved access processes are also accelerated and their retransmission. As standard feature extraction methodologies they operate mainly in the field of pixels, where digital data represent luminance intensities, it is not possible to directly apply them to compressed space without decompressing the visual information. The decompression requires an extra step that introduces delays and, in many cases, does not computationally advantageous. The most widely used standard for compressing visual information data today is the JPEG (Joint Photographic Experts Group) compression standard. The fundamental implementation of JPEG uses the DCT. The DCT having been exploited in many cases for image compression even in combination with other methods such as fractals, exploits as cosine basis functions of variable local frequencies and has been applied widely in the literature as an efficient way to extract features[2] [3][4].

A Compression using the DCT implementation of JPEG begins by separating each channel of images into segments of a fixed pixel size, usually a power of two per dimension, on which the DCT is applied creating spectral information which is represented by the derived coefficients. A typical segmentation for JPEG compressed images and MPEG video frames is done using 8×8-pixel segments. The spectral extraction of information with the DCT leads to concentrating most of the useful spectral energy over a small number of low frequency coefficients, for most of the images. The derived coefficients are then quantized and so the digital information is reduced by achieving compression. After quantization there is one more step in which the quantized coefficients are encoded based on their entropy using variable length Huffman codes which form the compressed image file. In general, DCT is applied using Equation (1) and (2) as follow :

$$F_{i,j} = C(u)C(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{i,j}\, cos\left[\frac{\pi(2x+1)u}{2N}\right] cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad (1)$$

where,

$$C(\kappa) = \begin{cases} \sqrt{\frac{1}{n}}, & \text{for } \kappa = 0 \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

the coefficients $F_{i,j}$ are the corresponding generated results for each spatial coordinate (u, v), where $u, v \in [0, N)$, of the part of the image. In total, $N \times N$ coefficients are produced and in a typical case where $N=8$, 64 coefficients are produced. Where $u=0$ and $v=0$ then the derivative coefficient is given as in as in Equation (3):

$$\frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{i,j} \quad (3)$$

which corresponds to the average value of the pixels of the entire processed segment image and that is why it is also referred to as direct current (DC) coefficient. The remaining $N \times N - 1$

coefficients they are also referred to as Alternating Current(AC) coefficients and contain the relevant spectral information of the processed image segment, in terms of frequency and directivity. "e.g.", the set of values $F_u \neq 0$, $v = 0$ contains spectral information with respect to horizontal axis of the image while the set of values $F_u = 0$, $v \neq 0$ with respect to the vertical axis.

As already mentioned, in most images the largest percentage of it of spectral energy is contained in the coefficients corresponding to low frequencies. The other coefficients contain information that can be excluded, with the zeroing of these coefficients without any significant effect on the relative visual information. The extraction, quantization and encoding of these coefficients ultimately leads to a compressed form of information. The reconstruction of visual information from compressed images using it above method, is done through the inverse discrete transformation cosine transform (IDCT). The compressed and encoded spectral information is first decoded and then converted into information pixel brightness via IDCT. The reconstructed images do not the information from the factors that have been excluded is now included, nor the parts of them that have been quantized during the compression process. However, this missing information in the majority of cases is not useful and does not affect, at least to some extent on a case-by-case basis, the fidelity of the original visual information. Based on the JPEG compression standard, an active research area has been established which focuses on methods of exploiting compressed information for issues search and avoiding the intermediate step of decompression. Algorithms have been developed in several research efforts extracting content features directly into the compressed space. the relationship between coefficients of the DCT from an 8x8 pixel segment and its 4×4 subsegments taking a smaller size image for individual feature extraction with the aim of searching for content. In later work to extract the features an algorithm was used which was based on the calculation of a set of statistical moments from the coefficients of parts of the DCT.

The DCT has been widely applied for efficient texture feature selection. It uses cosines of varying spatial frequencies as basis functions and is quite well known for its implementation in the JPEG compression standard. In texture images, enough of the signal information is contained in the low frequency components and appears in the upper left angle of the DCT. Knowing how the transformation transforms spatial information into frequency domain, texture features can be defined by its segment's spectral energy over a local region of the image. As the coefficients zero frequency (DC) represent almost the average monochromatic value of each part of the total $N \times N$ parts into which the respective image is segmented, it is considered to contain no texture information and is therefore ignored. The remaining coefficients of the DCT include the details, or otherwise the frequency and directional properties of the texture within a portion of the image and thus can considered to characterize the texture images and therefore can be used to extract texture features.

To extract such texture features, the images initially are partitioned into $N \times N$ partitions, with $N=16$ for the needs of the posterior's experiments. The size of the sections is chosen so that it is relatively reduced the number of extracted feature vectors but also at the same time texture information is effectively captured by using larger in size departments. Furthermore, experimentally this number has been confirmed to give better results classification results compared to smaller segment sizes, "e.g.", $N=8$. DCT is then applied separately to each segment and texture is represented as a feature vector V_m with $m \in [1,2\,N\text{-}2]$, the elements of which consist of the square roots of the sum of the coefficients at corresponding diagonals of the transformation. Considering that an image is segmented into total M segments of dimensions $N \times N$ then a set of M vectors of features can be extracted by this process which optimally represents the content of the texture images.

2.2 Convolutional Neural Networks

A CNNs is an artificial neural network (ANN) that makes use of deep mastering algorithms to analyze snap shots, classify visible elements, and perform laptop vision tasks. CNNs use ideas of linear algebra along with: B. Matrix multiplication for recognizing styles in pictures. These approaches involve complex computations, so a Graphics processing unit (GPU) is needed to educate the version. Simply positioned, CNNs use deep mastering algorithms to take enter data, such as pictures, and assign that means to aspects of that photo inside the shape of learnable biases and weights. This allows CNNs to differentiate or classify pixel [5]. A traditional CNN consists of the subsequent layers:

2.2.1 Convolutional Layer

Convolutional layers are the spine of CNNs. The parameters of this specific layer consist of a chain of learnable filters, also known as kernels, that have a totally small receptive variety, however which traverse the whole intensity of the enter volume. In the forward skip, every clear out is convolved over the peak and width of the enter quantity and sooner or later the inner product between these entries is discovered to supply the 2-Dimensional (2D) activation map for that particular clear out. Therefore, the network has a tendency to learn filters that set off when certain sorts of functions are detected at certain spatial region inputs. This gave us some key principles and terminology utilized in CNNs [5]:

2.2.1.1 Filters

Also known as feature detectors or kernels, filters can have specific dimensions, "e.g.", 2×2(it performs matrix multiplication per element by the input image to apply convolution).

2.2.1.2 Padding

Used to extend the input array to the edge of the array by inserting dummy pixels(This is done to counteract the fact that convolution reduces the size of the array, "e.g.", a 16×16 matrix can be transformed into a 4×4 matrix after filtering).

2.2.1.3 Strides

The stride is the number of pixels to shift the entire input matrix. If the increment is 1, we move the filter 1 pixel at a time, "e.g.", With a stride of 2, we shift each filter by 2 pixels, and so on.

2.2.1.4 Weights And Bias

A weight is a true price assigned to each enter/feature that conveys the importance of that specific function in predicting the very last output and bias is introduced to the made of the input and the load(is only a steady cost or constant vector Bias is used to balance the results Bias is used to shift the end result of the activation characteristic to the high-quality or terrible aspect)**.**

2.2.1.5 Rectified Linear Unit (ReLU*)*

Considered one of the few milestones in the deep mastering revolution. It is simple however extra powerful than its predecessor activation functions such as Sigmoid or Hyperbolic Tangent (Tanh)(The ReLU feature is its derivative and is monotonic and The characteristic returns 0 if given a negative enter, however for any positive cost of x it returns that cost and this effects in an output ranging from 0 to infinity).

2.2.1.6 Receptive Field

In a neural network, every neuron receives statistics from a one-of-a-kind factor inside the previous layer(in a convolutional layer, each neuron receives statistics from an area bounded only via the previous layer, known as the neuron's receptive discipline that In the case of a totally related layer, the entire layer above is the receptive field).

2.2.2 Pooling Layer

Paintings by dividing the enter characteristic map into a set of non-overlapping regions known as pooling regions that Each pooled vicinity is then converted right into a unmarried output fee, representing the presence of a specific feature in that region(the most not unusual sorts of pooling operations are max pooling and average pooling)**.**

2.2.2.1 Average Pooling

It maintains records approximately less critical features. It clearly scales down by way of dividing the input matrix into rectangular regions and computing the suggest of each location.

2.2.2.2 Max Pooling

A rule that takes the most price of a area and allows to hold the use of the most essential functions in the picture. This is a sample-based technique that converts a non-stop characteristic to a discrete one(its most important intention is to decrease the enter by using reducing dimensionality and making hypotheses approximately the functions contained in the rejected subregions).

2.2.3 Fully Connected Layer

It is one of the main categories of image classification and image recognition in neural networks. Scene labeling, object recognition, and face recognition are some of the areas where CNNs are widely used. CNNs consist of many layers, even hundreds of them. These layers learn to recognize different features of a given image[5]. CNNs route and transform inputs through many hidden layers. Each layer is created with a set of artificial neurons fully connected to all neurons in the same layer. Finally, there is a fully connected layer or output layer to display the results. CNNs, on the other hand, organize layers in three dimensions: width, depth, and height. Here, neurons in one layer connect only to a small area of neurons, rather than interacting with every neuron in the next layer. Finally, the result is represented by a single vector with probability values and only depth dimension. Each node layer has a threshold and a weight and is connected to each other. When the threshold is exceeded, the data is sent to the next layer of the network. These layers can perform operations to alter the data to learn related features. Furthermore, these operations iterate over hundreds of different layers that continuously learn to recognize other features of the image.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. RELATED WORKS

In 2017, Zeng et al. [6], proposed the first deep learning based steganalysis framework with a pre-processing block at enter recommended through Rich Models [7]. Proposed community can achieve exquisite performance increase as compared to Discrete Cosine Transform Residuals (DCTR) [8], however nevertheless not so good as Phase Aware Projection Model (PHARM). Xu-NetV3 et al. [9], Building Deep 20-Level Neural Communities for JPEG Network Analysis, Strongly Stimulated by Res-Net [10] with Link Hints. The proposed network with convolutional layers modifying the pooling layers also improves the results in terms of accuracy. It cut the mistake charge to 35% completed as in [11] for large scale JPEG steganalysis. Later, Chen et al. [12], introduce a deep gaining knowledge of framework with segment-break up concept stimulated by using JPEG compression set of rules. In their network they delivered two methods for incorporating phase focus in the community architecture, that is P-Net and V-Net. The experimental consequences exhibit that the proposed CNN shape is performing superior to Selection Channel Awareness-Gabor Filter Residuals (SCA-GFR) on Jpeg-Universal Wavelet Relative Distortion (J-UNIWARD) and Uniform Embedding Distortion (UED). Tsang, et al. [13], used a modified Ye-Net [14] without the choice channel-aware component. One of the changes is the addition of a BN (batch normalization) after each ReLU that would be useful to prevents the network from over fitting and offers a barely higher detection accuracy. Secondly, the writer reduces the stride of ninth convolutional layer before class to one, which made the scale of the 16 functions earlier than the Input padding (IP) layer to be 7×7 in preference to 3×3 as in the unique Ye-Net. Mo Chen et al. [15], proposed CNN based totally regressor for each jpeg and spatial picture steganalysis. The layout known as bucket estimator, starts offevolved by education a circle of relatives of CNN detectors, every detector for a hard and fast payload after which the usage of their concatenated function maps as a function on which a completely linked community (regressor) is train by way of the usage of the Mean Square Error (MSE) because the loss characteristic, the layout come out because the pleasant performer among different natural choices. The best CNN framework for JPEG as well as spatial steganalysis at the cease of 2018 is SR-Net that has been proposed with aspect-channel-information [16]. The network is like the aggregate of convolutional blocks past the pooling layer right away after the primary convolution block of the Yedrouj- Net [17]. Essential part of SR-Net is noise residual extraction section include first seven layers. The Network is very gradual as evaluate to in accordance [22] it takes 20 hours to educate the community, whilst SR-Net takes 1 week. HU DONGHUI et al. [2019], proposed a brand-new self-searching out steganalysis framework rely on deep reinforcement leaning & visible attention [18] to understand JPEG based adaptive steganographic algorithms. Their technique alternate photo into Allergic Fungal Rhinosinusitis (AFRs) through visual attention scheme after that makes repeated judgment by using reinforcement studying to pick AFRs which might be more suitable for steganalysis. Researchers finished sizeable improvement in detecting jpeg-based steganography algorithms via making use of extraordinary deep mastering-primarily based strategies of their frameworks. Yang-Net et al. [19], proposed a deeper 32-layers CNN framework with characteristic reuse approach through integrating all features from the prior layers as a end result enhance gradient & drift of statistics. Bottleneck layers and shared capabilities in their network similarly raise function propagation and decrease the version parameters dramatically. Experiment outcomes shows that the proposed structure can reduce the error fee 5.67% for 0.2 Bit Per Non-Zero Alternate Current (bpnzAC) and 4.41% for 0.4bpnzAC, while the number of education parameters in their framework is handiest 17% of what utilized by as in [9]. In 2017 Wu et al. [20], proposed deep residual framework that has two main variations with current community. First, proposed network is deeper than the prevailing networks which prove to be more productive to capture the statistical characteristic of digital photographs Secondly residual learning is used to actively preserve stego sign coming from stego pictures this is extraordinarily useful for fumigate of stego and cover images. One essential factor that we've got mentioned on this community is the batch normalization (BN) isn't used correctly. Huang et al. [2018], proposed a variation of as in [9] known as ResDet to stumble on adaptive JPEG steganography with close results. The network executed higher than as evaluate to in accordance [9], [12] with high embedding price. Zhong et al. [21], proposed any other a hit framework for steganalysis of jpeg photographs relying on clear out diversity phase. In their network the writer initiates three ensemble methods intended to increase the variety among classifiers. Xu-NetV1 et al. [22], proposed the primary deep learning framework achieved competitive overall

performance as compared with spatial rich model (SRM) [23]. In their proposed community, they used an absolute ABS activation layer for feature map generated from the first convolutional layer. It can examine greater resultful functions that might be helpful to avoid overfitting problems. They also used BN (batch normalization) and pooling layers of their community and finished higher accuracy to SRM [23]. Another milestone for jpeg photo steganalysis who've made huge contribution in step with [11], for hybrid deep neural community models. Proposed Network consists of two fundamental levels: The first segment is synthesized segment, analogous to the convolution & quantization truncation segment of SRM [24] and the secondary segment keep a composite neural community, which research parameters of the community throughout education technique. In their Network they also used three sub-nets stimulated in line with[22]. Proposed framework that first time install QT (quantization & truncation) into deep mastering based totally steganalysis. It is less green than Xu-NetV3 but provide first compelling technique reconcile with the aid of Rich models. Its performance is higher than DCTR, PHARM, Xu-NetV1 on J-UNIWARD, UED, Uniform Embedding Revisited Distortion (UERD). L. Gueguen et al. [25]. It changed into tested how DCT coefficients may be successfully used to educate CNNs for category, wherein the DCT coefficients can be computed or taken directly from JPEG photo layout. DCT-based CNN includes [26–28] and other DCT-based ResNet. The author uses the DCT parts of the JPEG images [26-27], but he doesn't use them to train the network because the images are stored in the DCT domain. The complication operation transfigures the sphere is only used for the test in [28]. The authors did not train CNN kernels in the DCT region. Just do their fast DCT calculations and reduce them by changing the 3×3 Conv2D planes to 2×2 boundaries. Our walkthrough uses a DCT sphere and a nonlinear soft threshold driver. Not suitable for training networks using DCT with or without soft thresholding.

4. PROPOSED METHODOLOGY

We introduced the subsequent operation: DCT of the feature maps generated via the convolutional layers of the ResNet-50. This step is executed after nonlinear thresholding of the convolutional functions, accompanied with the aid of the usual degrees of max pooling and next convolutional layers. We evaluated several methods of integrating DCT into the network and discovered that exceptional results had been acquired whilst acting simplest one DCT operation after the first convolution and thresholding steps. The regular effect of DCT when carried out to 8x8 blocks of photograph information is to concentrate most of the bigger values inside the pinnacle left block. However, in our version, we follow DCT to the characteristic maps produced by using the first convolutional layer inside the network. The features extracted from the image are threshold and then converted into the DCT area, followed with the aid of the standard pooling and convolution system inside the equal domain. Therefore, the next feature extraction after the primary convolutional layer is carried out within the frequency domain.

Our process is driven by reusing the well-working and widely used JPEG image compression. First, let's briefly explain how compression works. JPEG uses a YCbCr color scheme consisting of a luma component (Y) representing lightness and two chrominance components (Cb and Cr) that capture the color difference between blue and red. Color channels are sometimes subsampled for greater compression ratios. Each image channel is divided into 8x8 pixel blocks, each pixel value is subtracted by 128, and finally the region is transformed by a 2D DCT.

A 2D DCT transform (see Section 2.1) is applied to the input image patches. The result of the DCT transform is a patch image in the frequency domain. The top left corner of the patch contains low frequencies, while the bottom right contains high frequencies. If the compression is lossy, the patch coefficients are quantized by an 8x8 matrix to discard high frequency information.

The coefficients in the transform and quantization patches are arranged in anti-diagonal order starting from the upper left corner (lowest frequency). Finally, this one-dimensional sequence is compressed with run-length coding and converted into a Huffman code (entropy coding). Compression is described in more detail in Section 2.1. Given the non-homogeneous structure of Huffman codes, their variable lengths pose challenges for fitting data to fixed-size input networks, such as CNN-based architectures. Our experiments here focus on using the fixed-size output of the DCT transform as input to our classifier as shown in Figure (1).

The color information is important for reliable estimation of local features in digital images. Choosing an appropriate color space often plays an important role in tasks such as skin

segmentation. Most color space transformations are simple linear operations, and a CNN with sufficient capacity should be able to learn the transformation implicitly. We consider dividing an image in the YCbCr color space into blocks or windows defined by a window size t. Considering windows of any size, since JPEG compression divides the image into windows of size 8, instead of using the compressed data, each window is converted from the original image data to DCT space. Alternatively, if $t=8$ is desired, the entropy-encoded JPEG image can be decoded and dequantized to obtain input. This process is already performed when extracting RGB information from a JPEG image, and the extra computational cost of performing an inverse DCT can be avoided.

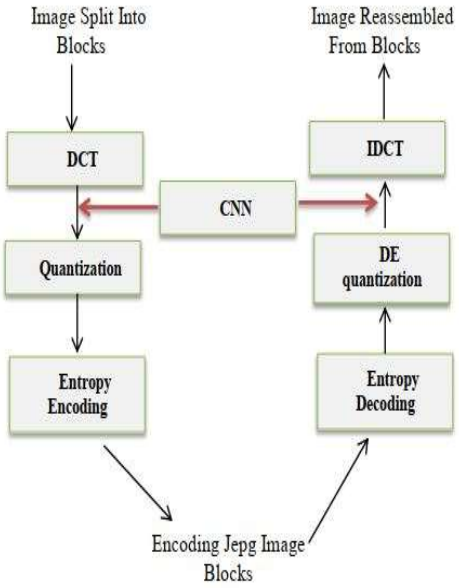

Figure 1: JPEG Decompression Steps: Red Arrows Indicate Where CNN Classifier Can Be Plugged-In.

The output of the DCT transform is a set of frequency coefficients, each given by a specific DCT basis function. The collection of these coefficients represents the feature space of the image window. The input filter $W \in \mathbb{R}^{Kt \times kt}$ of the proposed method is designed to cover k adjacent windows in each image direction. Filter application to an input image x in a block DCT space of block size $t \times t$ can be represented by the following cross-correlation defined on window coordinates $u, v \in N$ as in Equation (4) :

$$W * X(u\,t, v\,t) = \sum_{i=0}^{kt-1} \sum_{j=0}^{kt-1} w(i,j)\, x(u\,t + v\,t + j) \quad (4)$$

we will mainly use 2 window neighborhoods $(K=2)$ to form a $2t \times 2t$ kernel. Learnable filter weights can be viewed as a weighted average of the input DCT coefficients. The spatial position of a weight in the filter determines which frequency in the feature space it is responsible for. To avoid applying weights to coefficients with different frequencies, the convolutional filter slides along the image in steps of window size t. Stride is a step size in quantity of pixels by which the convolutional filter slides along rows and columns of an image as shown in the Figure (2).

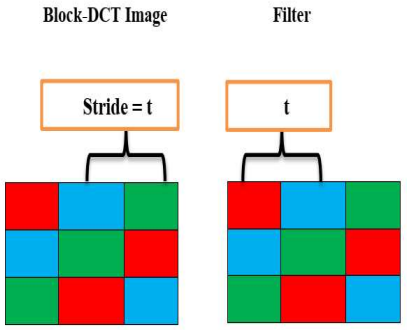

Figure 2 :Filtering Block-DCT Image With Window Size T = 2 And Filter Of Size 2t × 2t.

for window size t=8, a 16×16 filter is used. Note that the window does not need to be a square size when fed to a CNN, it can be of any shape consisting of t^2 values (e.g., a 1D vector $1 \times t^2$ as long as the relative positions of the windows remain constant). Other CNNs Layers do not require any modification and are applied directly on top of the first layer. The overall suggested procedure is shown in Figure (3).

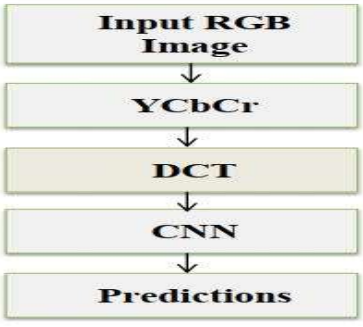

Figure 3 : Overall Suggested Model.

We first convert the image from RGB domain to YCbCr domain, then take blocks of size $N \times N$ and perform 2D DCT on these blocks using N^2 to obtain ordered frequency components. We replace $N \times N$ pixel blocks with equivalently transformed frequency coefficients. An $N \times N$ block requires N^2 time steps to perfectly reconstruct the block of pixels. A small value of N allows us to reconstruct the image by summing only a few basic frames. the JPEG compression standard makes use of an 8×8 DCT transform within the YCbCr color space, and the Cb and Cr additives provide $8 \times 8 = 64$ channels, one for each frequency, for a total of 192 channels. Assuming the original RGB enter photograph is of the form $H \times W \times C$, where $C = 3$, the peak and width of the photograph are denoted because the form of the H and W input features respectively After transforming to the frequency domain, it becomes $H/8 \times W/8 \times 64$ C , So the size of the input number has not changed. We bypass the stride-2 convolutional enter layer of the equal old CNN version, due to the fact the DCT The enter characteristic maps inside the region are smaller than their equal features inside the spatial region. If the max pooling operator straight away follows the enter convolution, we also bypass it. The channel length of the following layer is then fine-tuned according to the range of channels in the DCT domain. We take 64 channels, e.g., The three input layers of ResNet-50 are removed to produce a $65 \times 65 \times 64$ DCT input. We adjust the authentic CNN version in this manner to simply accept DCT features as enter. CNN fashions generally use $224 \times 224 \times 3$ enter facts in picture class tasks, which can be commonly down sampled from higher decision pictures. In ResNet50, the input DCT capabilities are sure to the primary the residual block, increasing the wide variety of channels to 192, ensuing in input capabilities of the form $65 \times 65 \times 192$. The DCT transform of the input image has dimensions of $448 \times 448 \times 3$, preserving four times as much information in the spatial domain as the corresponding $224 \times 224 \times 3$ input features[29].

The typical CNN body structure has many such convolution layers, and each layer has a different number of width and height filter. The nonlinear of the model can realize the activation function with the help of the activation layer, such as Sigmoid, TANH, and RELU [5]. Suppose that $K \in R^{m \times n \times C}$ is the filter in the convolution layer and the size is $M \times N \times C$. The size of $M \times N$ is the same as the size of each filter. C is the number of sewer channels. The size of the filter and the number of filters is the number and architecture selection of the network. The filter is called K. Given an input $I \in \mathbb{R}^{W \times H \times C}$ of the size $W \times H \times C$, the convolutional layer will lead to the output $Z \in \mathbb{R}^{\acute{W} \times \acute{H} \times \acute{C}}$, where $\acute{C}$ is the number of filters of $M \times N$ in the convolutional layer will lead to the output Z layer. In short, the convolutional layer will cause the following operations as in Equation (5) [5]:

$$Z^{\acute{c}}(x, y) = b^{\acute{c}} + \sum_{s=\frac{(m-1)}{2}}^{\frac{(m-1)}{2}} \sum_{t=\frac{(n-1)}{2}}^{\frac{(n-1)}{2}} \sum_{c=1}^{C} K_c^{\acute{c}}(s, t) I_c(x+s, y+t) \quad (5)$$

Where ,

$\forall \acute{c} = 1,2,\ldots, \acute{C} \quad \forall x = 1,2,\ldots, \acute{M} \quad \forall y = 1,2,\ldots, \acute{N}$.
$Z_c^{\acute{c}}$: $c^{'th}$ *channel of K for the* $c^{'th}$ *channel of the output Z*
$b^{\acute{c}}$ *:Bais for* $c^{'th}$ *channel the output G.*
I_c : $c^{'th}$ *channel the input I.*
$Z^{\acute{c}}$: $c^{'th}$ *channel the output Z.*

The pooling layer is performed through the network, it is used to reduce input. The largest pool is the popular choice of the pool level. One of the small windows with $N \times N$ size is scanned by input. The window selects the maximum pixel in the window. The replacement option of pooling layers is average pooling, minimum pooling, etc. These pooling layers are optional layers, adding again is the architectural choice.

In the CNN architecture, the output is usually directed by a Fully connected layer in the final stage of the convolution block. Completely connected layer execution of the same elements as standard ANN. At the end of CNN, there is a soft magnetic layer. CNN calculates the cost by calculating the input indicator as the mark, and then uses the optimization algorithm to understand the filter weight and distortion of the network, and then use the optimization algorithm to minimize the cost of the filter. Most of the time is also used to reduce the problem of more than adapting to data, while reducing the cost function.

A set of filters representing a convolutional layer is usually represented as a 4-dimensional tensor $V \in \mathbb{R}^{m \times n \times d \times d}$ for combining each of the n input feature maps with its own filter of size $d \times d$ Convolution, which means that the m output spans the feature map as in Equation (6) [30]:

$$y_i = \sum_{i=0}^{n-1} X * W_{ij} \quad (6)$$

many layers in newer architectures are so-called resampling layers, whose filters have a spatial extent of 1 ×1 ($d=1$), e.g., They just reevaluate existing traits. A fully connected layer can also be viewed as a special case of a $d=1$ convolutional layer that resamples features consisting of a single scalar. Our goal is to compress the weights of layers using a 1D DCT transform. The columns of the transformation matrix $C \in \mathbb{R}^{n \times n}$ represent the DCT basis functions with elements $C(a, u)$ as in Equation (7)(8)[30] :

$$C(a,u) = \sqrt{\frac{a(u)}{n} \cos\left[\frac{\pi}{n}\left(a+\frac{1}{2}\right)u\right]} \quad (7)$$

where,

$$a(u) = \begin{cases} 1, & \text{for } u=0 \\ 2, & \text{otherwise.} \end{cases} \quad (8)$$

Let an input matrix $V \in \mathbb{R}^{d \times d}$ corresponding to According to DCT, the matrix $v \in \mathbb{R}^{d \times d}$ in the frequency domain is defined as given in Equation (9),(10)[31]:

$$v_{j_1 j_2} = X_{j1} X_{j2} \sum_{i_1=0}^{d-1} \sum_{i_2=0}^{d-1} S(i_1, i_2, j_1, j_2) V_{i_1 i_2} \quad (9)$$

where,

$$S(i_1, i_2, j_1, j_2) = \cos\left[\frac{\pi}{d}\left(i_1+\frac{1}{2}\right)j_1\right] \cos\left[\frac{\pi}{d}\left(i_2+\frac{1}{2}\right)j_2\right] \quad (10)$$

is the cosine basis function, if $j=0$ then $X_j = \sqrt{\frac{1}{d}}$, otherwise $X_j = \sqrt{\frac{2}{d}}$. We use the abbreviation f_{dct} to refer to the DCT operation as in Equation (9) , $v = f_{dct}(V)$. The IDCT transforms v from the frequency domain back to the spatial domain and reconstructs V losslessly as given as in Equation (11) [31]:

$$v_{i_1 i_2} = X_{j_1} X_{j_2} \sum_{j_1=0}^{d-1} \sum_{j_2=0}^{d-1} X_{j1} X_{j2} C(i_1, i_2, j_1, j_2) V_{j_1 j_2} \quad (11)$$

we denote the IDCT function in Equation (11) it is $f_{dct}^{-1}(v)$, that is, $V = f_{dct}^{-1}(v)$. The matrix $V^{k\ell} \in \mathbb{R}^{d \times d}$ represents the weight matrix of $d \times d$ convolutional filters connecting the K^{th} input layer and ℓ^{th} output layer. (For simplicity, we assume quadratic filters and only consider filters in a single layer of the network). The weights of all filters inside the convolutional layer are represented by a 4-dimensional tensor $V \in \mathbb{R}^{m \times n \times d \times d}$, where m and n are the number of input layers and output layers respectively, and there is a total of $m \times n \times d^2$ parameters. Convolution filters can be represented equivalently in the spatial domain or in the frequency domain, with a DCT and its inverse mapping between the two. We denote the filter in the frequency domain as in Equation (12)[31]:

$$v^{k\ell} = f_{dct}(V^{k\ell}) \in \mathbb{R}^{d \times d} \quad (12)$$

and restore the original spatial representation as in Equation (13):

$$v^{k\ell} = f_{dct}^{-1}(V^{k\ell}) \in \mathbb{R}^{d \times d} \quad (13)$$

such as Equation. (10) or (11). The tensor of all filters is denoted by $v \in \mathbb{R}^{m \times n \times d \times d}$. Create training parameters (weights) for the DCT-convolution layer a fourth-rank tensor in $\mathbb{R}^{m \times n \times d \times d}$, where in m is filter out, n is the range of channels, $d \times d$ is the height or the width of the filter out. Filter is a series of n Slice $d \times d$ matrix. Performing IDCT on weights tensor creates a filter tensor of the equal form such that IDCT is finished independently on every slice. Once created, Tensors of filters are utilized in convolutional layers as ordinary. In some architectures, which include ResNet, the clear out $d = d = 1$. In fact, a layer of such filters is described as Matrix in $\mathbb{R}^{m \times n}$ (rank 2 tensor -1).

5. RESULTS AND DISCUSSION

Our experiments are carried out on a workstation computer with an NVIDIA RTX X6000 GPU. The code is written in PyTorch in Python 3.7. First, we experimented on the ImageNet-1K dataset. Then, we compare our model (DCT Feature based ResNet-50) with other related works on the ImageNet-1K dataset.

5.1 Classification Results On Imagenet

We show here the results obtained with the ImageNet 1K classification task. The ImageNet Visual recognition paradigms changed rapidly after the appearance of the ImageNet dataset, which demonstrated the power of data-driven feature learning. At this point, it is worth mentioning that the ImageNet dataset is a large-scale set, as it contains over 1.2 million physical images from 800 different classes [32]. In this paper, we use the official ImageNet 1K training code from PyTorch [34].

We use ResNet [33], that does not have a fixed depth and depends on the number of consecutive modules used. However, increasing the depth of the network to obtain a greater precision makes the network more difficult to optimize since it is more likely that the problem of disappearance occurs of gradients. ResNet addresses this problem by adjusting a residual application instead of the original and adding several connections between

layers. These new connections skip several layers and perform an identity or a *1×1* convolution. The base block of this network is called the residual block and is composed, when the network has 50 or more layers, by three sequential convolutions, one *1×1*, one *3×3*, and one *1×1*, and a connection that joins the input of the first convolution to the output of the third convolution, In our case we have use model ResNet-50 and replace *3×3* Conv2D layers in convolution blocks except for Conv2_1, Conv3_1, Conv4_1, and Conv5_1, as the down sampling is performed by them. Details of our DCT-ResNet 50, as presented in Table (1).

Table 1: Structure Of DCT-Resnet-50 For The Imagenet-1K Classification Task.

Layer Name	Output Shape	50-Layers
Conv_1 MaxPool	64× 112 × 112 64×56×5 6	7x7, 64, stride 2 3×3 stride 2
Conv2_1	256×56×56	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64, \text{stride } 2 \\ 1\times1, & 265 \end{bmatrix}$
Conv2_x	256×56×56	$\begin{bmatrix} 1\times1, & 64 \\ DCT, & 64 \\ 1\times1, & 265 \end{bmatrix}\times2$
Conv3_1	512×28×28	$\begin{bmatrix} 1\times1, & 128 \\ 3\times3, & 128, \text{stride } 2 \\ 1\times1, & 512 \end{bmatrix}$
Conv3_x	512×28×28	$\begin{bmatrix} 1\times1, & 128 \\ DCT, & 128 \\ 1\times1, & 512 \end{bmatrix}\times3$
Conv4_1	1024×14×14	$\begin{bmatrix} 1\times1, & 256 \\ 3\times3, & 256, \text{stride } 2 \\ 1\times1, & 1024 \end{bmatrix}$
Conv4_x	1024×14×14	$\begin{bmatrix} 1\times1, & 256 \\ DCT, & 256 \\ 1\times1, & 1024 \end{bmatrix}\times5$
Conv5_1	2048×7×7	$\begin{bmatrix} 1\times1, & 512 \\ 3\times3, & 512, \text{stride } 2 \\ 1\times1, & 2048 \end{bmatrix}$
Conv5_x	2048×7×7	$\begin{bmatrix} 1\times1, & 512 \\ DCT, & 512 \\ 1\times1, & 2048 \end{bmatrix}\times2$
GAP	2048	Global average pooling
Output	1000	Linear

We compare the performance of the implemented in [25], [26], [27], [28], [34], and [35] models and the proposed model. Our experiments are performed on a workstation computer equipped with an NVIDIA RTX X6000 GPU. Code is written in PyTorch in Python 3.7. First, let's experiment with the ImageNet 1K dataset. We then compare the ResNet-50 model with other related work on the ImageNet 1K dataset. In this section, We use NVIDIA RTX X6000 to train ResNet-50 and corresponding DCT version. Standard training requires about 10-15GB of GPU memory, while the RTX X6000 only has 24GB. Therefore, we halve the batch size and learning rate accordingly. We use the Stochastic gradient descent (SGD) optimizer with a weight drop of 0.0001 and a momentum of 0.9. DCT-based ResNet-50 is trained with default settings: The mini-batch size is 128, and the initial learning rate is 0.05 for 90 epochs. After every 30 epochs, the learning rate is reduced by 1/10. For data inference, we apply a random size crop to the training images to obtain 224 x 224 images, and then randomly flip the images horizontally. We normalize the images using a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225], respectively. We evaluate our models on the ImageNet 1K validation dataset and compare them to state-of-the-art papers. During training, the best models based on center crop top (1) accuracy are stored in the ImageNet 1K validation dataset, and their accuracy numbers are reported in Table(2). In Figure(4), shows the process of testing errors in the ImageNet 1K validation dataset during the training phase. Compared with the base ResNet-50 model. A DCT-ResNet-50 model contains 28.5% fewer parameters, 32.8% less Multiply accumulate (MAC), and its top-1 center crop accuracy drops from only 76.06% to 75.22%. Compared with the base ResNet-50 model. In Figure(5), shows the process of testing errors in the ImageNet 1K validation dataset during the training phase. Compared with the base ResNet-50 model. A DCT(24-channel)-ResNet-50 model contains 21.1% fewer parameters and 32.6% fewer MACs, while the top-1 average cut accuracy only drops from 76.06% to 75.62%. In Figure(6), shows the process of testing errors in the ImageNet 1K validation dataset during the training phase. Compared with the base ResNet-50 model. A DCT(64-channel)-ResNet-50 model contains 4.7% fewer parameters and 32.5% fewer MACs than the base ResNet-50 model, while center-crop pre-1 accuracy drops from only 76.06% to 75.72%.

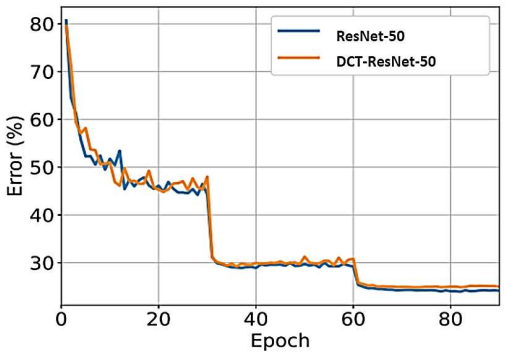

Figure 4 : Training On Imagenet-1K. Curves Denote The Validation Error Of The Center Crops. Comparison With Resnet-50 Vs Our DCT-Resnet-50.

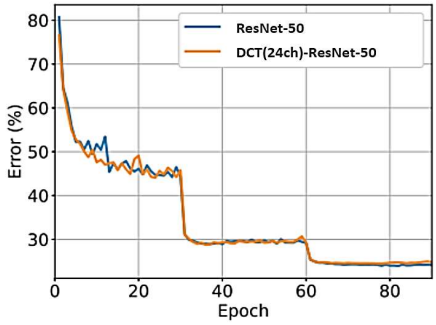

Figure 5 : Training On Imagenet-1K. Curves Denote The Validation Error Of The Center Crops. Comparison With Resnet-50 Vs Our DCT(24-channel)-Resnet-50.

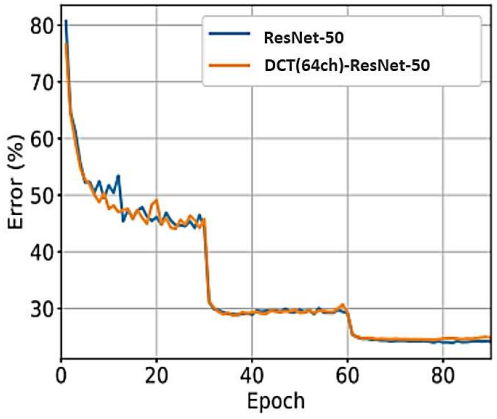

Figure 6 : Training On Imagenet-1K. Curves Denote The Validation Error Of The Center Crops. Comparison With Resnet-50 Vs Our DCT(64-channel)-Resnet-50.

Table 2 : Classification Errors On Imagenet Validation Set Using Central Crops

Model	*Parameters*	*MACs (G)*	*TOP-1 %*	*TOP-5 %*
JPEG-ResNet-50 [25]	28.4	5.4	76.06%	93.02%
ResNet-50+DCT+FBS(3x32) [26]	26.2	3.68	70.22%	-
ResNet-50+DCT+FBS(3x16) [26]	25.6	3.18	67.03%	-
Faster JPEG-ResNet-50 [27]	25.1	2.86	70.49%	
DCT-based ResNet-50 [28]	21.30	-	74.73%	92.30%
ResNet-50[33](offical [34])	25.65	3.68	75.3%	92.2%
ResNet-50[33](Torchvision [35])	25.65	4.122	76.13%	92.86%
ResNet-50 (our trial, baseline)	25.56	4.122	76.06%	92.85%
DCT-ResNet-50	**18.28 (28.5%)**	**2.772 (32.8%)**	**75.22%**	**92.52%**
DCT(24-channel) - ResNet-50	**20.15 (21.1%)**	**2.780 (32.6%)**	**75.62%**	**92.61%**
DCT(64-channel) - ResNet-50	**24.35 (4.7%)**	**2.783 (32.5%)**	**75.72%**	**92.71%**

Able to moreover increment the exactness of normal ResNet-50 by embedding DCT with batch normalization after the global average pooling layer of ResNet-50. This leads to a irrelevant increment in MACs. As appeared in Table 3, this increments the center-crop top-1 precision from 77.53% to 77.67% as it were with 2.3% additional parameters with 0.05% higher MACs, and it increments center-crop top-5 exactness from 92.75% to 93.84% with 16.4% additional parameters with 0.05% higher MACs.

Table 3 : ImageNet-1K results of inserting DCT before the global average pooling layer.

Model	*Parameters*	*MACs (G)*	*TOP-1 %*	*TOP-5 %*
ResNet-50	25.65	4.122	77.53 %	92.75 %
DCT-ResNet-50	**29.76**	**4.124**	**77.67 %**	**93.84 %**
DCT(24-channel) - ResNet-50	**31.17**	**4.132**	**77.82%**	**93.92%**
DCT(64-channel) - ResNet-50	**35.740**	**4.133**	**77.95%**	**93.93%**

6. CONCLUSIONS

In this paper, we propose and test our idea to change the input representation to train the existing ResNet-50. We introduced the subsequent operation: DCT of the feature maps generated via the convolutional layers of the ResNet-50. This step is executed after nonlinear thresholding of the convolutional functions, accompanied with the aid of the usual degrees of max pooling and next convolutional layers. We evaluated several methods of integrating DCT into the network and discovered that exceptional results had been acquired whilst acting simplest one DCT operation after the first convolution and thresholding steps. Our proposed DCT function based on the standard ResNet-50 convolution operation, which is composed of the first Convolutional layers are generated for image classification problems. First, the color image is changed over from the RGB variety space to the YCbCr variety space, and afterward the R, G, B, and Y parts are parted into covering fixed-size blocks, and the highlights of the R, G, and B part pictures are made. From one viewpoint, from R, Blocks are extricated from DCT portrayals of G, B, and Y part picture endlessly obstructs are separated from them then again. The proposed model shows that ResNet can also be trained using JPEG-compressed DCT coefficients and subsequently perform better compared to traditional ResNet methods. Testing the Efficiency of Modified Input Representations on ImageNet-1K Using Existing ResNet-50 Architecture and Proposed DCT-Resnet-50 Architecture. A DCT-ResNet-50 model contains 28.5% fewer parameters, 32.8% less MAC, and its top-1 center crop accuracy drops from only 76.06% to 75.22%. Compared with the base ResNet-50 model.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Graduate Studies at Jouf University for funding and supporting this research through the initiative of DGS, Graduate Students Research Support (GSR) at Jouf University, Saudi Arabia.

REFERENCES:

[1] O. Garcia, J. Sandoval, N.Miyatake and P.Meana, “Color image steganalysis method for LSB matching,” Proceedings of the International Conference on Security and Management (SAM), the Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Las Vegas, Nevada, USA, July 27–30, 2015, pp. 27–30.

[2] V.Senthooran,L.Ranathunga“DCT Coefficient Dependent Quantization Table Modification Steganographic Algorithm,” IEEE network softcomputing(ICNSC) 2014 international conference, Aug. 2014, pp.19-50.

[3] N. Ponomarenko, V. Lukin, K. Egiazarian, J. Astola, “DCT based high quality image compression,” Scandinavian Conference on Image Analysis, 2005, pp. 1177-1200.

[4] A.V. Bazhyna, N.N. Ponomarenko, K.O. Egiazarian, V.V. Lukin, “Efficient Scalable DCT Block-based Image Coder with Compression of Signs of DCT Coefficients,” Telecommunications and Radio Engineering, Vol. 67 (5), 2008, pp. 391-412.

[5] H. H. Aghdam and E. J. Heravi “Guide to Convolutional Neural Networks : A Practical Application to Traffic-Sign ,” 2017, [Online]. Available: https://www.pdfdrive.com/guide-to-convolutional-neural-networks-a-practical-application-to-traffic-sign-detection-and-classification-d188544301.html.

[6] J. Zeng, S. Tan, B. Li, and J. Huang, “Pre-training via fitting deep neural network to rich-model features extraction procedure and its effect on deep learning for steganalysis,” Electronic Imaging, vol. 2017, no. 7, 2017, pp. 44–49.

[7] S. Baluja, “Hiding images in plain sight: Deep steganography,” Advances in Neural Information Processing Systems, 2017, pp. 2069– 2079.

[8] V. Holub and J. Fridrich, "Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT," IEEE Transactions on Information Forensics and Security, vol. 10, no. 2 Feb 2015, pp. 219-228.

[9] G. Xu, “Deep convolutional neural network to detect J-UNIWARD,” in Proc. of the 5th ACM Workshop on Information Hiding and Multimedia Security. ACM, 2017, pp. 67–73.

[10] M. Zheng, S.-h. Zhong, S. Wu, and J. Jiang, “Steganographer detection via deep residual network,” in Proc. of 2017 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2017, pp.235–240.

[11] J. Zeng, S. Tan, B. Li, and J. Huang, “Large-scale JPEG image steganalysis using hybrid deep-learning framework,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, 2018, , pp. 1200–1214.

[12] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich, "Jpeg-phase- aware convolutional neural network for steganalysis of jpeg images," in Proc. of the 5th ACM Workshop on Information Hiding and Multimedia Security. ACM, 2017, pp. 75–84.

[13] C. F. Tsang and J. Fridrich, "Steganalyzing images of arbitrary size with CNNs," Electronic Imaging, vol. 2018, no. 7, 2018, pp. 1–8.

[14] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," IEEE Transactions on Information Forensics and Security, vol. 12, no. 11 , 2017, pp. 2545–2557.

[15] M. Chen, M. Boroumand, and J. Fridrich, "Deep learning regressors for quantitative steganalysis" , Electronic Imaging, vol. 2018, pp. 160-1-160-7(7).

[16] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images" , IEEE Transactions on Information Forensics and Security, vol. 14, no. 5,2019, pp. 1181–1193.

[17] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-Net: An efficient CNN for spatial steganalysis" , in Proc. of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 2092–2096.

[18] D. Hu, S. Zhou, Q. Shen, S. Zheng, "Digital image steganalysis based on visual attention and deep reinforcement learning", IEEE Access, 2019.

[19] J. Yang, Y.-Q. Shi, E. K. Wong, and X. Kang, "JPEG steganalysis based on Dense-Net," Multimeida, ArXiv abs/1711.09335, 2017.

[20] S. Wu, S. Zhong, "Deep residual learning for image steganalysis," Multimedia tools & Applications, vol. 77, no. 9, 2018, pp.10 437–10 453.

[21] K. Zhong, G. Feng, "Deep learning for steganalysis based on filter diversity selection," Science China Information Sciences, vol. 61, p. 129105, 2018.

[22] G. Xu, H.-Z. Wu, "Structural design of convolutional neural networks for steganalysis," IEEE Signal Processing Letters, vol. 23,2016, pp.708–712.

[23] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," IEEE Transactions on information Forensics and Security, vol. 7, no. 3, 2012, pp. 868–882.

[24] S. Baluja, "Hiding images in plain sight: Deep steganography," Advances in Neural Information Processing Systems, 2017, pp. 2069– 2079.

[25] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, J. Yosinski, "Faster neural networks straight from JPEG,Advances in Neural Information Processing Systems," 2018, pp. 3933-3944.

[26] S. F. D. Santos, N. Sebe, and J. Almeida," The good, the bad, and the ugly: Neural networks straight from jpeg," IEEE International Conference on Image Processing (ICIP), 2020, pp. 1896–1900.

[27] S. F. D. Santos and J. Almeida, "Less is more: Accelerating faster neural networks straight from jpeg. In Iberoamerican Congress on Pattern Recognition, Springer," 2021, pp. 237–247.

[28] Y. Xu, and H. Nakayama, "Dct-based fast spectral convolution for deep convolutional neural networks," International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8.

[29] K. Xu, M. Qin, F. Sun, Y. Wang, F. Ren , and Yen-Kuang Chen, , "Learning in the Frequency Domain," ArXiv abs/ 2002.12416, 2020.

[30] M. Ulicny, V. A. Krylov, R. Dahyot, "Tensor Reordering for CNN Compression," ArXiv abs/1506.04449, 2020.

[31] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing Convolutional Neural Networks in the Frequency Domain," ArXiv abs/1506.04449, 2015.

[32] J. K. Olga Russakovsky, Jia Deng, Hao Su, A. Sanjeev Satheesh, Sean Ma, Zhiheng Huang, and et al. Karpathy, Aditya Khosla, Michael Bernstein, "Imagenet large scale visual recognition challenge.," Inter- Natl. J. Comput. Vision, Vol. 115, No.2, 2015, pp. 211–252.

[33] K. He, X. Zhang, S. Ren, and Jian Sun, "Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition," 2016, pp. 770–778.

[34] Deep residual networks official code. https://github.com / KaimingHe / deep - residual - networks,2015. Accessed: 29/03/2023.

[35] Models and pre-trained weights. https://pytorch org/vision/stable/models.html, 2023. Accessed: 29/03/2023.