IMAGE FORGERY DETECTION USING COLOR-CORRELATION ANALYSIS OR CONVOLUTIONAL NEURAL NETWORKS

Linglong Le, Ha Nguyen

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May. 24, 2021

Technical Report No. ECE-2021-02

Contents

1 Introduction

A growing concern in everyday life is the detection of image forgeries to determine whether an image has been digitally modified. As image modification software such as Photoshop has become increasingly sophisticated, with more and more advanced techniques being used to modify and alter images for various purposes, it has become even more important to develop techniques to detect these alterations. Several techniques for the detection of image forgery have been presented in the past, such as the use of watermarking as the media is being recorded to mark it as trustworthy, but this requires the use of specialized equipment. A much more popular alternative is the application of various passive techniques, ranging from the detection of abnormalities in the statistical data of the image, to analyzing the underlying JPEG compression data of an image, to developing machine learning based processes to detect image tampering. To this end, the goal is to develop a method to apply two such passive techniques, a Convolutional Neural Network (CNN) and a technique centered around Color Filter Array (CFA) interpolation to detect tampered images. This project has been completed as one of the requirements in ECE course "Digital Image Processing and Communication" (EC520).

2 Literature Review

In [1], one such passive method based on CFA interpolated images was proposed. When an image is interpolated using a CFA, the samples naturally have a correlation due to the nature of CFA, and forgery techniques destroy these correlations. This correlation is due to how color images in cameras are sampled using lattices, before CFA interpolation is used to generate the full image, which creates correlation between a pixel, and its neighboring pixels. Knowing this, the Expectation Maximization Algorithm is applied to the image, creating the probability map for the image, which captures this correlation. Then by taking the Fourier Transform of localized probability maps, a graph that can detect correlation is obtained. A lack of localized peaks in an area of the map indicates the absence of CFA correlation, which indicates that the image was tampered with.

In [2] a less computational intensive method based on [1] was proposed to detect image forgery. Here the interpolation coefficients would be roughly estimated, and more focus would be placed on the obtained probability map. The probability data was then transformed into a 1-D data set using a sigmoid scan, which greatly simplifies the calculations. The Fourier Transform is then applied to this data, and used to calculate a Frequency Characteristic Value (FCV). A threshold value can then be calculated for the value of pi in this FCV graph, then compared with the actual data. If the threshold is significantly exceeded by what is obtained at $f = \pi$, then the image is genuine, and has not experienced tampering.

A more advanced method requiring optimization and deep learning was introduced in [3]. This method does not leverage CFA Interpolation, and instead utilizes a

Convolutional Neural Network (CNN) to make decisions based on full-resolution information gathered from the whole image. This model includes dividing input images into overlapping patches and then extracting discriminative features. To do so, high pass image residuals called image noiseprints were extracted from the image being analyzed, to emphasize camera related artifacts.These image noiseprints were then fed into the network along with the image color bands, as these noiseprints will highlight possible spatial anomalies and may help in detecting local manipulations. Next, feature aggregation is performed to add some form of pooling, maximum, minimum, and average to the model. Finally, after aggregating the local information into a single descriptor for the whole image, decisions are made using a few fully-connected layers for classification, to determine if an image was tampered.

3 Problem Statement

Given these prior implementations and approaches to the problem of image forgery, two main approaches to detecting forgeries were selected, a CFA based method and a CNN based method. The CFA method will utilize the work described in [2] to detect tampering by leveraging the correlation created by CFA interpolation. However, to understand this process, an understanding of the EM Algorithm and CFA Interpolation is necessary, thus a section describing these two concepts is included in the appendix. This method assumes that the images given are images which have preserved the correlation created by CFA Interpolation, and that tampering is applied after CFA Interpolation. Then the portions within an image that have distorted correlation from CFA Interpolation are identified, and marked as tampered areas. To do so, first probability maps must be generated from the input image using the Expectation Maximization(EM) Algorithm, which can capture the correlations created by the CFA. Then these maps are transformed into 1D vectors, before the Fourier Transform is applied to them to create FCV graphs, which identify correlation through the presence of a Peak at $f = \pi$. To detect if a peak is present, a threshold function is then introduced. If the threshold is exceeded then a peak is present, thus, if the threshold is not exceeded at $f = \pi$ for a given area, then it has been tampered with. This general process is described by the block diagram in Figure 1

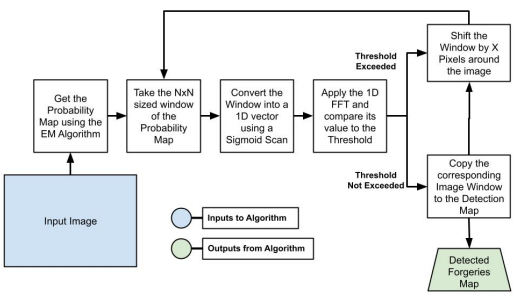


Figure 1: Proposed Solution using CFA Interpolation

In conjunction with this approach, a neural network-based approach will be applied, inheriting the ideas from the CNN-based end-to-end method proposed in [3]. Due to computational and memory bottlenecks, deep networks will only accept small sized images as input, for example 256 x 256 pixel images. Instead of resizing the inputs to fit the network model, the framework will instead divide the image into multiple overlapping patches, to extract distinguishing features with the desired sizes. Each patch was assigned a label for supervised training. For this project, some pre-trained networks such as VGG16 and ResNet50 were tested and compared, before the one with better performance was selected. The output from this constructed network is the label of forged or unforged for the extracted patch from the image. To detect forgery from a large sized image using this model, the image is first split into random patches with the correct size, then fed into the network to predict the label for each patch. Combining these local decisions for the patches together, it is then possible to classify the image as a whole. This approach is shown in Figure 2.

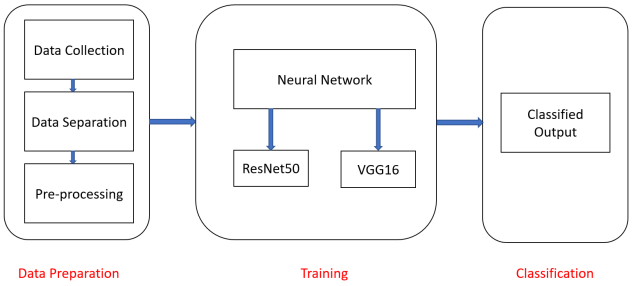


Figure 2: Proposed solution for image forgery detection using CNN

4 Implementation of CFA Based Method

4.1 Implementation of the EM Algorithm

The EM Algorithm implemented here is introduced in [2], which fixes the interpolation coefficients used to generate the error function, and creates one error map that will be used throughout the rest of the algorithm. The parameter that is being maximized here is the standard deviation of these probability maps, which are modeled using the standard Gaussian probability model.

To begin the EM algorithm, several parameters must first be initialized. These parameters are the initial standard deviation σ_i, the constant ρ_0, and the termination value η. The standard deviation σ_i is used to initialize EM algorithm, while the termination value η is the parameter that indicates when the EM algorithm should terminate. The constant ρ_0 is used for normalization necessary to execute the Maximization Step. For the EM implementation introduced by [2], σ_i is initialized to be 30, ρ_0 is initialized to be $1/256$, and η is set to be 0.001.

In addition to those variables, the CFA interpolation coefficients must also be chosen for the calculation of the error map, which will be integral to the EM steps. For this implementation of the EM Algorithm, the coefficients are taken to be both known, and fixed to a basic Bayer CFA Pattern, as opposed to optimized in each cycle. This is because the main goal of this algorithm is to detect correlations from CFA Interpolation, instead of being very accurate. However, for use in the EM algorithm, the set of coefficients is different from the normal CFA coefficient matrix, as the interest is not in creating an CFA interpolated image, but measuring how far a given image is from one that was CFA interpolated, by recovering the original sample values. These matrices are shown below, with further explanation on the selection of values included in the appendix:

$$\alpha_g = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix} \qquad \alpha_r = \alpha_b = \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix}$$

α_g represents the coefficients for the green channel error calculation, while α_r and α_b represent the coefficients for the red and blue channels error calculations respectively. This calculation is performed over all the pixels in a given image, except for the outer edge due to boundary concerns, and can be expressed as follows:

$$e_g(x,y) = \left| I_g(x,y) - \sum_{u,v=-1}^{1} \alpha_g(u,v) \cdot I_g(x-u, y-v) \right| \tag{1}$$

where $e_g(x,y)$ is the error map for green, which is generated by taking $I_g(x,y)$, or the value of the image's green channel, and subtracting from it the reconstructed values that represent CFA interpolation. These reconstructed values are obtained by convolving $\alpha_g(u,v)$, with the green channel value. The error map generated by equation (1) describes how similar the input image's green channel is to one that was generated using the fixed coefficients assumed, with darker areas being areas that are close to CFA Interpolation, and lighter areas being different from CFA interpolation. To generate the red or blue channels error maps, the image values and error coefficients above would need to be swapped to their red or blue counterparts. An example set of error maps are shown in Figure 17, located in the appendix.

Having generated the error map, the EM algorithm can then begin to generate probability maps for the image. First, the Expectation step is performed, by approximating the probability distribution for the image using the error map and standard deviation. For this algorithm, the probability distribution is modelled as a Gaussian distribution as follows:

$$P(x,y) = \frac{1}{\sigma^{n-1}\sqrt{2\pi}} exp\left(-\frac{e^2(x,y)}{2[\sigma^{n-1}]^2}\right) \tag{2}$$

where $e^2(x,y)$ is the error map squared and σ^{n-1} is the standard deviation from before the current iteration of the Expectation step. This process takes the error map and previous standard deviation to calculate a new probability map, $P(x,y)$. This newly

obtained probability map can then be used to calculate a new standard deviation that would correspond to the current iteration, thus labeled σ^n. But before doing this, the probability map must be normalized, to bring the standard deviations into a consistent range, so that the termination value does not need to be fine tuned for each new image. However, this is a simple process as shown below.

$$w(x,y) = \frac{P(x,y)}{P(x,y) + \rho_0} \tag{3}$$

Here $w(x,y)$ is the normalized probability, based off of dividing the probability $P(x,y)$ by its own values plus the constant initialized in the first step. Using this normalized probability, the Maximization step can then be applied to calculate the new standard deviation, as shown in equation (4).

$$\sigma^n = \left[\frac{\sum_{x,y} w(x,y) e^2(x,y)}{\sum_{x,y} w(x,y)} \right]^{\frac{1}{2}} \tag{4}$$

This process of iterating through equations (2), (3) and (4) to find and maximize the standard deviation continues on until the difference in σ^n and σ^{n-1} is smaller than the termination value η set initially in the algorithm. Once this condition is reached, the algorithm exits, and the probability map generated in this iteration is set to be the probability map for the given image. The complete EM Algorithm implemented here was adapted from [2] and can be found in the appendix, in Figure 16.

4.2 Detection of CFA Interpolation

One key trait of the CFA Interpolation is that high values of correlation between a pixel and its neighbouring will only occur at the locations where the pixels was created from the interpolation. This correlation will thus occur periodically, as there are locations where the pixels values are directly from the sensor, and locations where the interpolation was used to fill in the missing values. This periodicity of correlation will also create periodicity within the probability map values as well, with higher probability values occurring at the pixels created from the CFA Interpolation. Thus, if an image has preserved the correlation from CFA interpolation, the resulting magnitude plot from applying a 2D Discrete Fourier Transform (DFT) to its probability map will have discrete valued peaks. This is due to the trait of the Fourier Transform that causes periodic data to result in discrete valued spectrum. These spectrum can then be reviewed by the human eye to determine whether or not any form of tampering has occurred, by detecting any irregularities within the peaks of the spectrum. An example of this shown in Figure 18 in the appendix.

However, if a specific type of scanning is used to convert the 2D probability map into a 1D vector, then the periodicity and correlation captured by the data is still preserved. In Figure 3 the scanning adapted from [2] used to convert the data from 2D to 1D is shown.

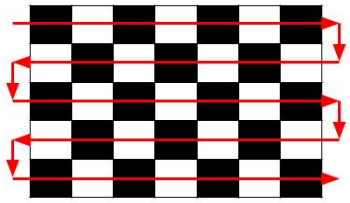
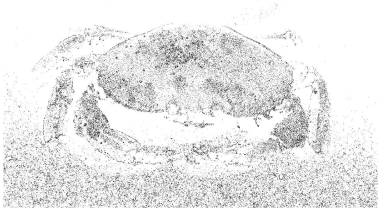
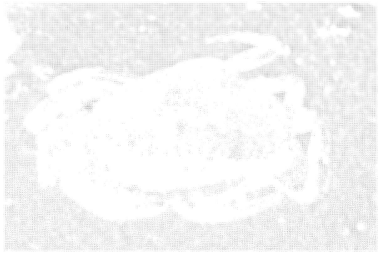


Figure 3: Scan from 2D to 1D

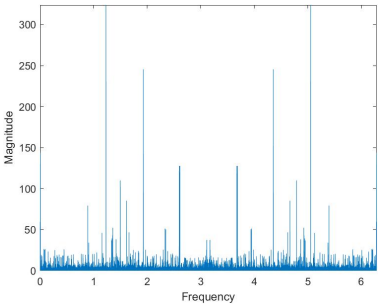
Within the 1D vector of values, there will also be a periodicity within the vector in the form of higher and lower areas of probability due to the nature of the correlation between pixels from the CFA Interpolation. This rapid swap between localized high and low values in the data will result in peaks at high frequencies when the 1D DFT is applied. Specifically, for this implementation the peaks will occur at $f = \pi$ as the DFT will result in a spectrum that has frequency values ranging from $[-\pi,\pi]$. So detecting a peak at $f = \pi$ in the 1D DFT indicates that the correlation from CFA Interpolation is present within the image. This property is showcased in Figure 4. For the purposes of this project, the frequency values from $[-\pi,0]$ will be shifted over to the range of $[\pi,2\pi]$, and the value at DC was removed. This is because the DC value is massive and would overshadow the rest of the data.



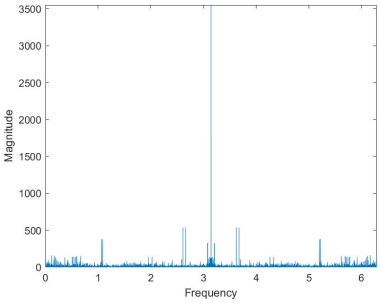
(a) Probability Map without Correlation



(b) Probability Map with Correlation



(c) 1D DFT Magnitude Plot of (a)



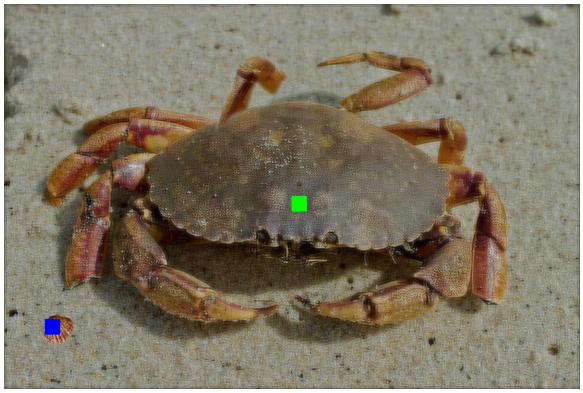
(d) 1D DFT Magnitude Plot of (b)

Figure 4: Probability Maps and the Corresponding DFT Magnitude Plots

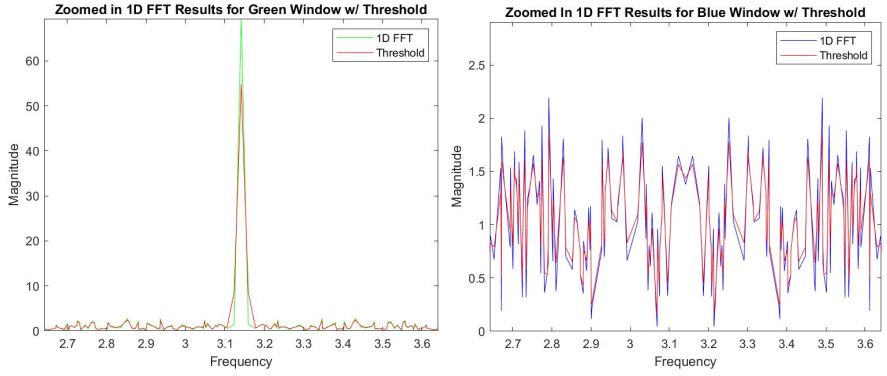
So with the 1D method, detecting a peak at the frequency of π is one way to determine if the correlation from CFA interpolation in the data has been preserved, or if it has been altered via tampering. This peak can be verified by eye, but a threshold

function can be used to automate this detection. As introduced in [2], performing a convolution between the 1D DFT data and an appropriately sized Gaussian will create such a threshold function. This is because the convolution assigns a value at each frequency that is the weighted sum of the values around that frequency. Thus for areas in the data that are large peaks, the result from the threshold will be lower compared to the values from the data. So if the threshold function is exceeded by the data at the frequency of π, then a peak is present, indicating that the correlation from CFA interpolation has been preserved. This method does not require the human eye, which means that the detection of CFA Correlation can be completely automated. Additionally, this method can be applied to smaller windows within the map, allowing the detection of localized regions of tampering.

To show this behavior, a CFA Interpolated image was tampered via a rotate and shift, and an insert of a new item into the image. The probability map was generated using the EM algorithm, then the process of obtaining the 1D DFT data and threshold data was applied. The results are show in Figure 5. Clearly it is possible to detect image tampering by obtaining the local probability map of a window, taking the 1D DFT, then comparing its value at $f = \pi$ to a threshold function.



(a) Image with Windows (Blue=Tampered, Green=Untampered)



(b) Untampered Window 1D FFT (c) Tampered Window 1D FFT

Figure 5: DFT and Threshold Plots for Tampered and Untampered Windows

5 Implementation of CNN Based Method

5.1 Data Preparation

For the CNN, the dataset was obtained from [4]. These images are either 1000x700 or 700x1000 in size, and are divided into three categories:

- Original images: 70 images that have not been manipulated.
- Fake images: 970 images with copy-move forgery.
- Mask images: 970 black and white images describing the copy-move parts of the corresponding fake images.

The images in the dataset are all simple scenes with a single object against a regular background. Some of the included fake images also have rotation and scaling applied to the copy-move forgery, so that the CNN can learn more about the boundary of tempered parts in different orientations as well. One example of a typical image set from the dataset is shown in Figure 6. For the purposes of this project, the entire dataset of original images and fake images was split into 70% training data to train the model up, and 30% testing data to test the model after it has been trained.

Figure 6: Example of a Set of Images from the Dataset.

For the training data, specific patches were then extracted from the fake images for use in training, using the following procedure:

1. Extract patches of size 64x64, with a stride length of 32px.
2. For each patch, count the number of white pixels in the corresponding mask image.
3. Each extracted patch to be used must consist of at least 20% white pixels and 20% black pixels, to ensure that the boundary of the forgery is captured.
4. Compare the extracted patch from the forgery to the corresponding patch from the original image, as these patches must be different.
5. Save the extracted patch if all the necessary conditions have been matched.

The dataset being used is small, thus it is desirable to generate as many patches as possible from each image. If a larger patch size is chosen, then less patches will be extracted thus limiting the amount of training data. But on the other hand, a patch size that is too small will fail to describe anything meaningful. Therefore for this project, a patch size of 64x64 was selected as a compromise to extract a significant amount of forged patches, while still maintaining enough meaningful data in each patch for the model to train on. Additionally, each forged patch must maintain at least 20% white and 20% black on the extracted mask patch to ensure that the extracted patches always contain a boundary between a forged and unforged region. However, since the mask has white pixels on both the original area and the forged area, an additional step must be taken to compare the extracted patch with the corresponding patch from the original image. If they are identical, then the extracted patch does not contain a forgery boundary, and thus must be discarded. An example of some forged patches that were extracted is shown in Figure 7.

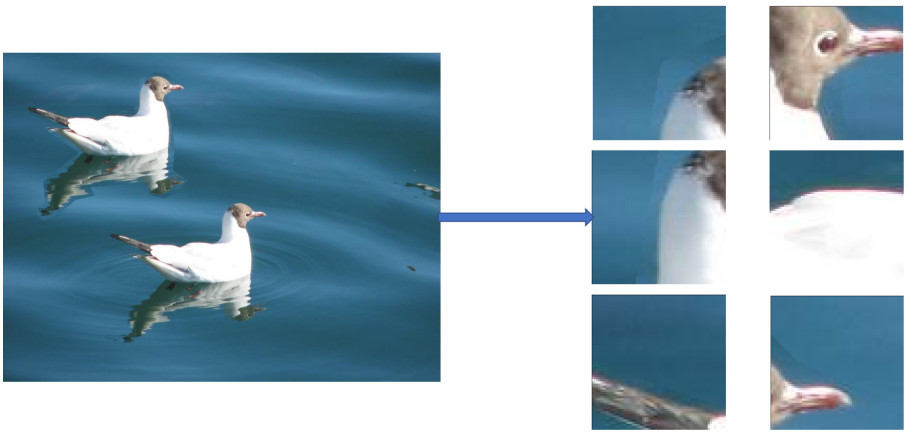


Figure 7: Example of Patches Extracted from a Fake Image.

After doing patch extraction on the forged data, around 15 forged patches were extracted from each fake image. However, to train the model, the amount of unforged patches should match the amount of forged patches. Thus from each original image, around 200 random unforged patches are extracted. After obtaining these patches, they were also split into 70% training and 30% validation. The distribution of data after undergoing this method is shown in Table 1.

Initial Data

	Training	Testing
Original	49	21
Fake	679	291

Patches Extracted

	Training	Testing
Original	9702	4200
Fake	9659	4184

Table 1: Data from initial images and after patch extraction

5.2 CNN Architecture

For the CNN model, the weights of a pretrained model from training on a much larger dataset were used to initialize the model. In this case, both VGG16 and ResNet50 were selected for the initialization, as they had been trained on the ImageNet dataset using millions of images. The weights from these networks were used as the initial weights for the model, and then further refined using the dataset generated above to obtain new weights for the model. Taking ideas from [5], the fine-tuning process starts with the weights from a trained network, then re-training on a new dataset using very small weight updates. Following this concept, the last convolution layer of the pretrained model was fine tuned using the procedure below:

1. Instantiate the convolutional base of the pretrained network and load its weights
2. Add the fully-connected model on top, and load its weights
3. Freeze the layers of the network model up to the last convolutional block.

This method was implemented on both ResNet50 and VGG16, but the results of initial testing showed that ResNet50 had a better performance, so it's weights were selected. The results of this initial testing for both networks are shown in the experimental results sections. Figure 8 shows the architecture used for the fine tuning process for ResNet50.

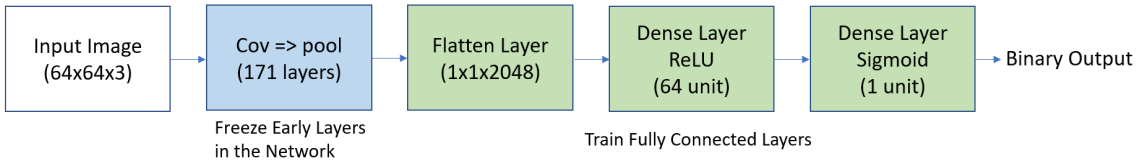


Figure 8: Fine-tuning the ResNet50 Architecture.

In [5], it was also suggested to use a SGD optimizer for the fine-tuning process, but it was found that applying the Adam optimizer yielded better performance for the model. To ensure that the previously learned features are not completely destroyed in the process of fine-tuning, the learning rate was 0.0001, so that the new weight is updated slowly. This training took about 90 minutes over 15 epochs on a machine with an Intel(R) Core i7-1065G7 CPU 1.5GHz, and 16GB RAM.

6 Experimental Results

6.1 Results from the CFA Method

To test the CFA Method, first a data set for testing was created. This data set consists of 10 JPEG Images of varying sizes that were downloaded from the internet. These images were down-sampled, then up-sampled by a factor of 2, before a filter was applied to them to mimic correlation that would be created using CFA interpolation. These images then had various types of digital tampering applied to them via GIMP such as insertions, rotations, scaling, blurring, and copy shift tampers. This resulted in a set of 12 tampered images, as some image had multiple tampers applied to them. These tampered images were then fed into the algorithm outlined in Figure 1, which was implemented in MATLAB. Shown below in Figures 9 and 10 are some selected image sets that showcase the strengths and shortcomings of this algorithm.

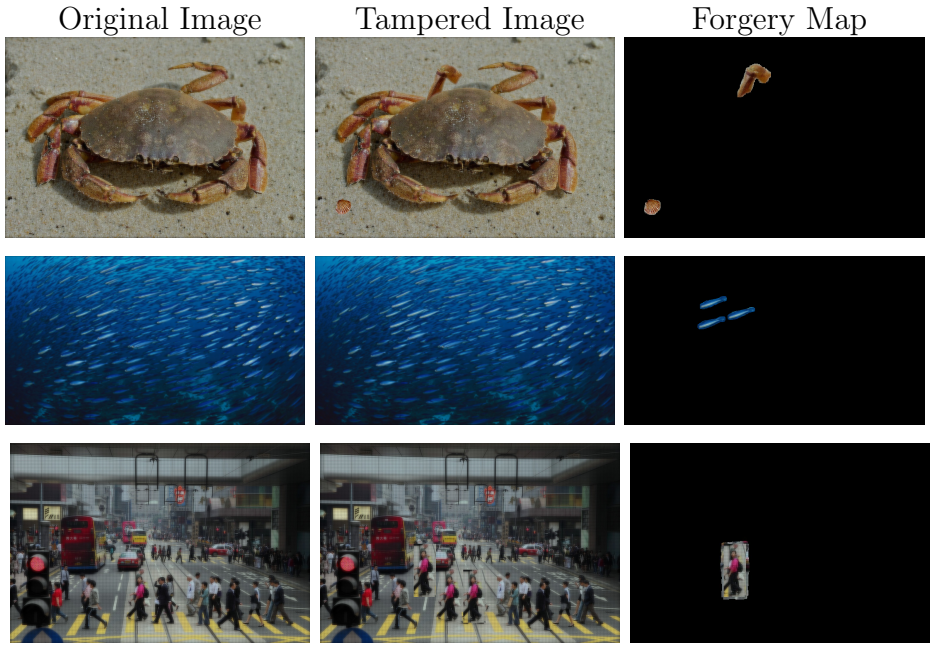


Figure 9: Results of Applying Algorithm with a 16x16 Window, Shift of 2

As can be seen in Figure 9, all of the tampered images fed into the algorithm successfully detected a tamper, although not all areas of tampering were captured, regardless of window size and window shift. However, there was one type of tampers that the algorithm performs relatively poorly with, the rectangular copy and shift tamper. This is due to the assumptions made in the generation of the probability map. Since the probability map is being generated using only the first order neighborhood of pixels for interpolation, and the window is rectangular in shape, the rectangular copy and shift will have a chance of preserving the relative periodicity within the probability map. This will maintain a peak at π in the 1D DFT Data, even if the image was tampered with, thus fooling the algorithm.

Figure 10: Effects of Window Size and Shift Size

Window and shift size also significantly affected the performance of the algorithm as shown in Figure 10. With larger window and shift sizes, the finer detailed tampers that occurred in the image were not caught by the algorithm. What is not shown however, is that the smaller window sizes and shifts sometimes identify small areas inside of the tampered as not tampered, while small areas outside of the tampered area are identified as tampered. However, despite these shortcomings, the results manage to show clearly which areas have been tampered and which areas are untampered, and give anyone looking at this image a clear sense of what regions are suspicious.

6.2 Results from the CNN Method

Figure 11 and Figure 12 show the model accuracy and losses over 15 epochs of applying the training process for the models initialized with ResNet50 and VGG16. For VGG16, the training accuracy reached a maximum 0.95 and the validation reached up to 0.85. However, for ResNet50 the training accuracy reached a maximum of 0.99, with a validation accuracy of up to 0.95. This led to the decision to proceed with the ResNet50 weights. However, for both models, while the training accuracy increased with epochs, the validation accuracy was erratic as the epoch changed. The same observation was also made for the loss, with training loss decreasing over time, while the validation loss jumped up and down. This may be because the model is overfitting, as the training accuracy is very high while the validation accuracy is unstable. If this is the case, then the issue can be solved by simply getting more training data, or by lowering the capacity of the model to memorize the training data. Another issue that occurred is the potential overlap of detected tampers. Since each fake image generates

around 15 corresponding fake patches, when extracting the forgeries from the fake images, some may overlap. To solve this, a method could be developed to extract relevant patches of the fake image without overlapping patches. Another option is to apply regularization to the model, but this will add a cost to the loss function of the network for large weights, and yield a simpler model that has only learned relevant patterns from the training data.

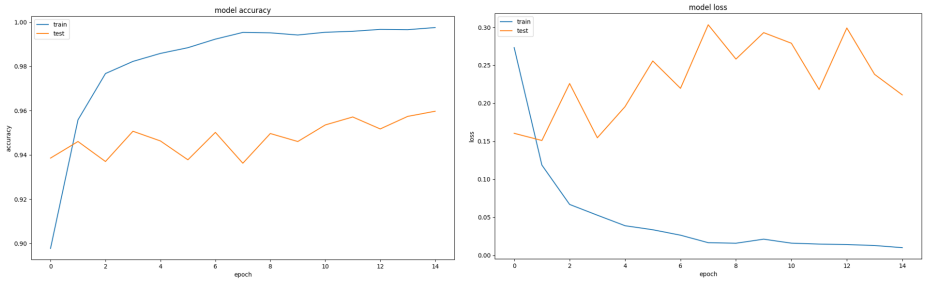


Figure 11: Model Accuracy and Loss for Training and Validation of ResNet50 Model.

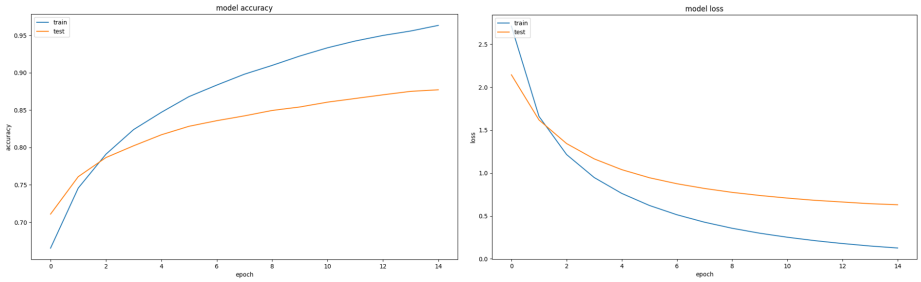
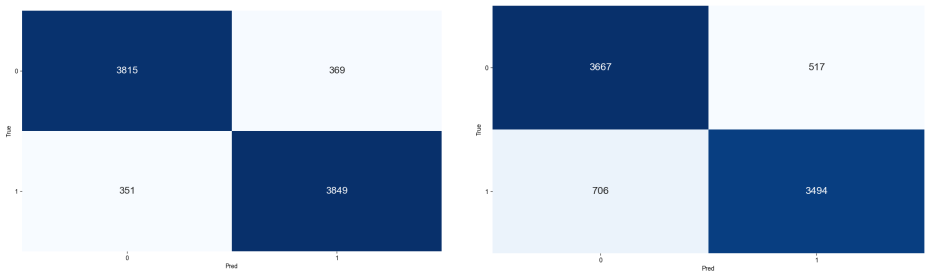


Figure 12: Model Accuracy and Loss for Training and Validation of VGG16 model.

For the results in this section, the same process for pre-processing data applied to the training data was applied to the data assigned to the test set. From this procedure, 4200 unforged patches and 4184 forged patches were generated for testing the model, as shown in Table 1. Then these patches were sent into the trained model, to yield the results shown in Figure 13. In these confusion matrices, a 0 represents the label for unforged patches, while 1 is the label for forged patches. Again, ResNet50 has much better performance with 91% of both forged and unforged patches being correctly labelled, compared to the 85% accuracy rate from the VGG16 based model.



ResNet50 Model VGG16 Model

Figure 13: Confusion Matrix from Testing the Models.

To assess performance of the model, first the whole test set was classified. Then the False Positive Rate (FPR) and True Positive Rate (TPR) were computed as functions of the classification threshold, going from 0 to 1, to obtain the corresponding Receiver Operating Characteristic (ROC) curve. The area under the ROC curve (AUC) was also computed, and assigned as a synthetic measure of performance. In addition to this, the precision-recall curve was also generated to show the trade-off between precision and recall for different classification thresholds. As shown in Figure 14, the AUC for ROC curve of ResNet50 network is higher than VGG16 network, while the AUC for Precision-recall curve is 0.94 for both network models. Thus it can be seen that overall, ResNet50 has better performance.

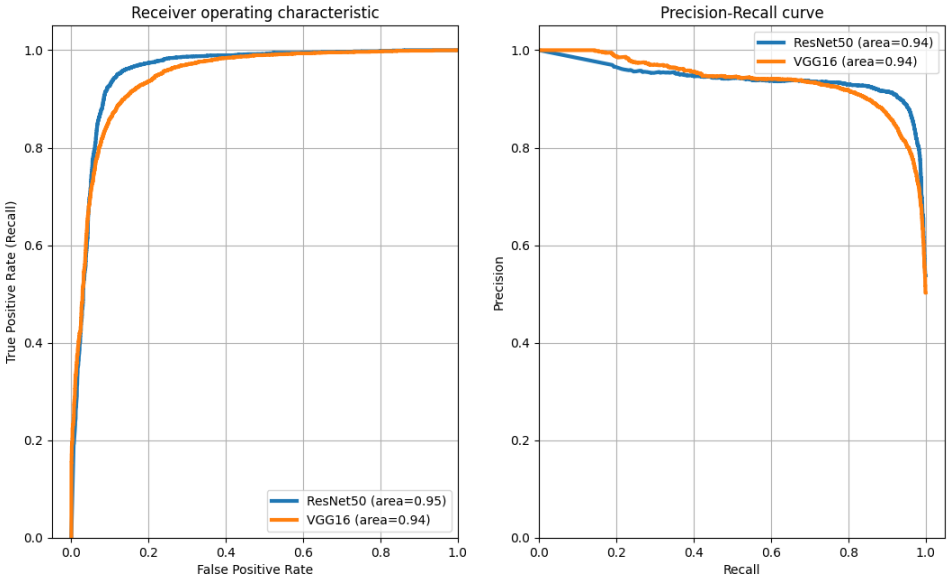


Figure 14: ROC and Precision-Recall curves for the models.

7 Conclusion

7.1 CFA Method Conclusion

In general, the CFA based algorithm developed is very good when it comes to detecting most types of tampering done to an image, and can adjust the window and shift size to adjust the level of detail needed in the forgery map. For images that have preserved the correlation from interpolation such as .RAW or .TIFF files, this method is extremely useful in seeing if any forms of tampering have been applied. However, this method only works for images that have this CFA generated correlation preserved within their data, and distorted by tampering. With JPEG files, or files that have had their correlations destroyed, this method will fail as it will register the entire image as being a tampered image. This method will also fail on images that have had tampering performed before a type of CFA interpolation is applied. Additionally this method is susceptible to rectangular copy and shifts, due to the assumptions made in order to simplify the calculation of the probability map.

One possible improvement that can be made to this method to resolve the issue of rectangular copy and shifts is to increase the order of the neighbourhood used in the calculation of the probability maps. This can be done by approximating a larger interpolation filter that is used within the image, but will significantly increase the computational complexity of the error map generation step. However, since this method is mainly concerned with utilizing correlation created through CFA interpolation, there is no way to improve on this method to work on JPEG Images or images that have had CFA interpolation applied after tampering.

7.2 CNN Method Conclusion

In conclusion, the proposed CNN method is good for images with smaller sizes that can fit into the input of the pretrained model. For larger images however, the predicted results from the model will only apply as a local decision for a region, not to the whole images. For example, if an original image of size 1000x2000 is fed into the system to detect tampering, with the current method method the image has to be split into hundreds of smaller patches of size 64x64. If about 100 of these smaller patches from this image are selected, then the model will predict that 91 of these images are unforged and 9 of them are forged, even though the entire image is unforged. Combining these predictions together, the model would incorrectly label the input image as tampered due to the 9 false positives that detected tampers.

One possible improvement to the model used in the CNN is to solve the overfitting problem, by either using a larger dataset for training, or by creating a more optimized method for patch extraction. Another option is to try a method such as the one proposed in [3], where instead of training based on extracted patches, these patches can be leveraged to learn about and classify the whole image, so that whole images can be used to train the model.

8 References

[1] A. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, 2005.

[2] T. Kuo, Y. Lo and S. Huang, "Image forensics utilizing color filter array periodic characteristics." National Taipei University of Technology, 2011.

[3] F. Marra, D. Gragnaniello, L. Verdoliva, and G. Poggi, "A full-image full-resolution end-to-end-trainable cnn framework for image forgery detection," *IEEE Access*, vol. 8, pp. 133488–133502, 2020.

[4] C. Group, "Copy-move forgery dataset." http://www.diid.unipa.it/cvip.

[5] F. Chollet, "Building powerful image classification models using very little data." https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html, 2016.

9 Appendix

9.1 Brief Overview of CFA Interpolation

One common method of capturing colored images is through the use of Color Filter Arrays, which will pass through only one of 3 colors (Red, Green, or Blue) to the sensor, to create 3 different lattices that have the specific color values on them. These lattices are then interpolated, to create 3 images that capture the R,G and B values of the image respectively, then merged to create a color image. An example of what these lattices look like can be seen in Figure 15.

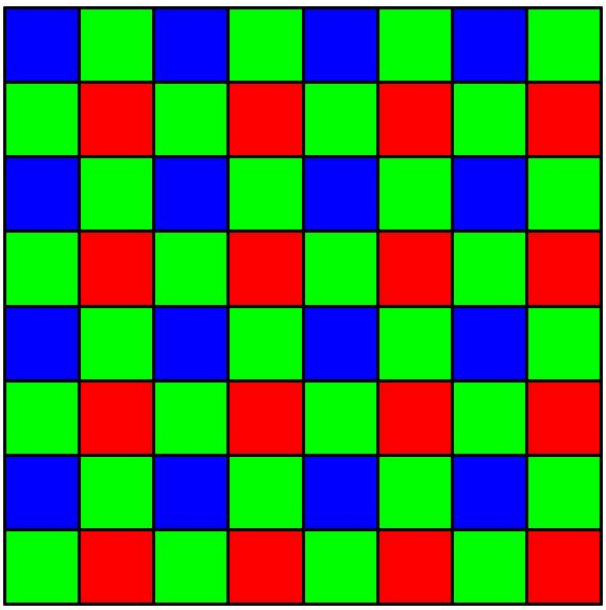


Figure 15: Example CFA Lattice

Since the values of pixels not on the lattice are dependent on the values from the applied interpolation, the pixels will be correlated with their neighbours. This correlation can then be detected on a probability map of the image, and leveraged to detect any tampering. If the image has been tampered with, then the correlation from the CFA interpolation will be destroyed or distorted.

9.2 Brief Overview of the EM Algorithm

The Expectation Maximization Algorithm is one of the main methods to generate joint probability distributions for a wide range of problems, where not all the variables are known. These unknown variables are called Latent Variables, and greatly complicate any attempts to generate probabilities, as there is no clear way to account and predict these Latent Variables.

The EM Algorithm is split up into 2 parts, the expectation step, and the maximization step. The expectation step attempts to calculate the expected value of these

Latent Variables by fixing a set parameters and then applying calculations to find expectation. The maximization step then attempts to find a new set of parameters that will maximize the value of these expected values for the latent variables.

Each maximization generates a new set of parameters that will be used in the Expectation step to calculate the new expected values. Once there is a small enough difference between the parameters generated before and after the maximization step, the EM algorithm exits, having generated a proper probability distribution for the data. By doing so, a relatively accurate probability model can be generated even for data sets without any accompanying information about the variables that control the values in the data set.

9.3 Explanation of the Error Matrix Coefficients

As shown earlier, in the EM algorithm the following matrices are used for the expectation step:

$$\alpha_g = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix} \qquad \alpha_r = \alpha_b = \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix}$$

This is based on the assumption that the CFA Interpolation coefficients are set up in the following manner.

$$c_g = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 1 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix} \qquad c_r = c_b = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

For the error calculation matrices α_x, it is desirable that the matrix is able to reconstruct the center pixel given the neighbourhood. As a result, the matrix should follow the general coefficients given by c_x, while maintaining a sum of 1, to avoid any extra intensity changes. For the creation of α_g, it is enough to simply remove the 1 in the center of c_g, to yield $alpha_g$ that will reconstruct the center value pixel based on the neighbourhood, assuming CFA interpolation occured. This is due to how the green sampling locations are spaced out on the Bayer Mosaic, so that each neighbouring sample location will only contribute to one pixel in the neighbourhood. However, for α_b and α_r, the presence of $\frac{1}{4}$ in the corners of complicates things. Simply removing the 1 in the center, and scaling the remaining coefficients so that the sum is one will almost certainly yield the wrong center value, as the corner pixels are created using 4 different sample locations, as compared to the 1 neighbouring sample location used at all other locations in the matrix. Thus, to counteract this, the $\frac{1}{4}$ in the corners can instead be inverted to become $-\frac{1}{4}$. This inversion will then thus reduce the influence of the neighbouring samples, as the corner pixels are mainly made up of neighbouring samples, instead of the sample being looked for, while also maintaining the sum of 1. Thus by having negatives in the corners, while retaining the rest of c_g, with the exception of the 1 in the center, the obtained sample pixel value will be much closer to the sample value used in the interpolation.

9.4 Full Expectation Maximization Algorithm

/*Initialize*/
choose initial σ_0=30;
set p_0=1/256;
set the iteration terminated condition η=0.001;
n=0;
/*error function*/
for (each pixel location (x,y)){

$$e(x,y)=\left|I(x,y)-\sum_{u,v=-1}^{1}\alpha_{u,v}\cdot I(x+u,y+v)\right|;$$

}
Do{
/*expectation step*/
for (each pixel location (x,y)){
/*likelihood*/

$$P(x,y)=\frac{1}{\sigma^{(n-1)}\sqrt{2\pi}}\exp\left(-\frac{e^2(x,y)}{2\left[\sigma^{(n-1)}\right]^2}\right);$$

/*CFA interpolation posterior probability*/

$$w(x,y)=\frac{P(x,y)}{P(x,y)+p_0};$$

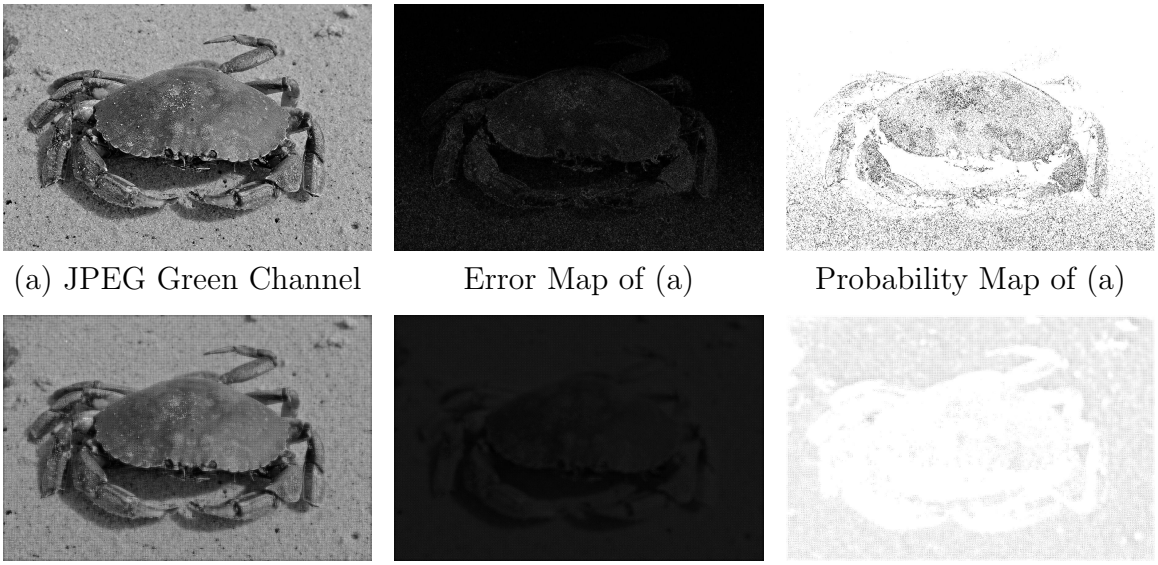
}
/*maximization step*/

$$\sigma^{(n)}=\left[\frac{\sum_{x,y}w(x,y)e^2(x,y)}{\sum_{x,y}w(x,y)}\right]^{\frac{1}{2}};$$

}**while**($|\sigma^{(n)}-\sigma^{(n-1)}|>\eta$)

Figure 16: EM Algorithm Implementation from [2]

9.5 Further Explanation on Error and Probability Figures

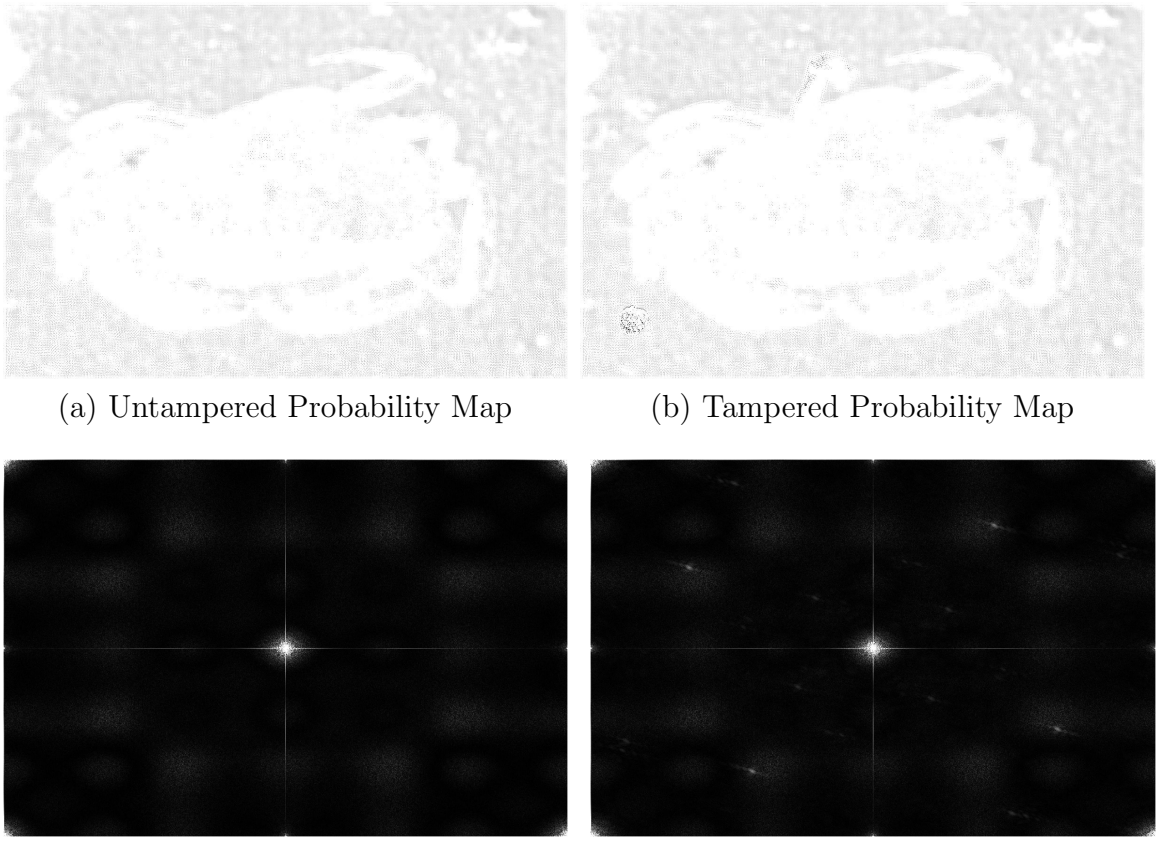


(a) JPEG Green Channel | Error Map of (a) | Probability Map of (a)

(b) Interpolated Green Channel | Error Map of (b) | Probability Map of (b)

Figure 17: Error Map Generated using Green Channel of an Image

As can be seen here, when the error map is applied to an image without any CFA interpolation applied, the areas of major change are highlighted, while areas of of little change are dark. However, with CFA Interpolation, the error map appears to simply be a darker version of the image itself. This shows that for the interpolated map, each pixel is closely related to its neighbourhood, while the uninterpolated map only shows close relation in areas of consistent color. This can then be tied to how the probability maps appear in 4, which have also been included in the figure above for convenience. Referring back to the model used to generate the probability maps (2), it can be seen that the value in the exponent for the Gaussian model is $-e^2(x, y)$. Thus areas of high error will result in low probability, and areas of low error will result in high probability, which is exactly what can be seen in figure 17 above. For the results of the JPEG image, the bright spots on the crab's shell resulted in dark areas on the corresponding probability map. Thus for the interpolated image, it can be seen that the shell of the crab is relatively darker compared to the surrounding sand for the error map. Due to this, the crab will appear much brighter as compared to the background, due to the generated error map.



(a) Untampered Probability Map (b) Tampered Probability Map

(c) Magnitude of 2D DFT of (a) (d) Magnitude of 2D DFT of (b)

Figure 18: Probability Maps and the Corresponding DFT Magnitude Plots

For the DFT Plots, max intensity was set to be the average of the max value in each row, due to how large the value at DC and near DC were. This simply causes areas of the image to max out in intensity, however the general behaviour still supports the argument.