CS 754
Course Project

Aakriti 190050002
Danish Angural 190050028

February 2021

Contents

Introduction

Computer graphics rendering software is capable of generating highly photorealistic images that can be impossible to differentiate from photographic images. As a result, the unique stature of photographs as a definitive recording of events is being diminished (the ease with which digital images can be manipulated is, of course, also contributing to this demise). Photorealistic computer generated graphics can be forged as photographic images, causing serious security problems.

The aim of our project is to use a deep neural network to detect photographic images (PI) versus photorealistic computer generated graphics (CG).

The acquisition of a photographic image (PI) requires photoelectric conversion, interpolation, and post-processing of the sensor to add noise to the image. CG images, on the other hand, are produced by software and have a *smoother* texture than PI.

Datasets

- 25k computer generated images were downloaded from https://www.kaggle.com/zarkonium/synt image-dataset-cats-dogs-bikes-cars
- 40k photorealistic images were downloaded from http://images.cocodataset.org/zips/unlabeled20

Motivation for first technique

Wavelet coefficients of natural images, that is, images obtained from a camera, are known to follow a laplacian distribution, that is, the coefficients are centered around 0 and the probability decreases steeply as one gets further.

$$P(x) = \frac{1}{Z} e^{-|\frac{x}{s}|^p}$$

This has previously been used for compressing images. However, in this implementation, we have tried to reflect the difference in the distributions of wavelet decomposition coefficients of photographic and photorealistic images, and on these basis, distinguish between them. Also, we use the distributions of the second and third level coefficients.

Approach

The image is first split into RGB bands for evaluation of i level wavelet decomposition.

Deciding the features

As mentioned before, we have used the first 4 statistics (mean, variance, skewness, kurtosis) for each wavelet sub-band at every level and every color band, which makes a total of $36(i-1)$ features.

As mentioned in the paper, this is not enough for a classifier to distinguish between a Computer generated and a photographic image. There also exist properties that relate the coefficients within the same subbands and other subbands. For example,

- a strong coefficient at a point would imply strong coefficients in all neighbouring points.
- a strong coefficient at a point at subband level i would imply a strong coefficient at level $i+1$
- also, there must exist a relation between the coefficients of the red, green, blue bands at the same level.

$$\begin{aligned}|V_i^g(x,y)| &= w_1|V_i^g(x1,y)| + w_2|V_i^g(x+1,y)| + w_3|V_i^g(x,y1)| \\ &+ w_4|V_i^g(x,y+1)| + w_5|V_{i+1}^g(x/2,y/2)| + w_6|D_i^g(x,y)| \\ &+ w_7|D_i^g + 1(x/2,y/2)| + w_8|V_i^r(x,y)| + w_9|V_i^b(x,y)|\end{aligned}$$

$$\begin{aligned}V_i^g &= Qw \\ p &= log(\frac{V_i^g}{Q(Q^TQ)^{-1}Q^TV_i^g})\end{aligned}$$

Using these properties, we calculate an error estimate, and add its mean, variance, skewness and kurtosis to our features.
Thus for an i level decomposition, we get 72(i-1) features. In our implementation, we have used a 4 level decomposition.

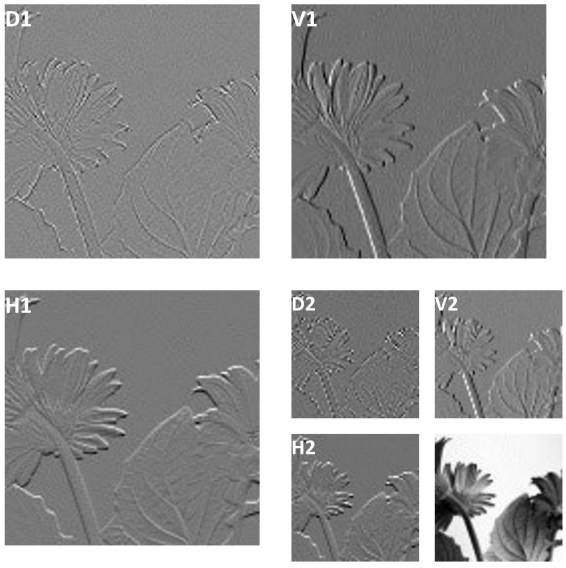


Figure 1: Magnitude of a multi-scale and orientation decomposition of a photographic image

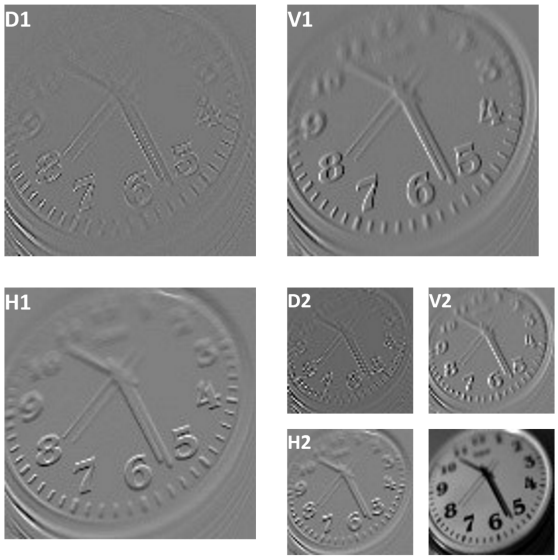


Figure 2: Magnitude of a multi-scale and orientation decomposition of a photorealistic computer graphic

Filters Used

As, mentioned in the paper, the following filter is found to be more efficient for classifying computer generated and photographic images.

$$l = [0.02807382, 0.060944743, 0.073386624, 0.41472545, 0.7973934, 0.41472545, 0.073386624, 0.060944743, 0.02807382]$$

$$h = [0.02807382, 0.060944743, 0.073386624, -0.41472545, 0.7973934, -0.41472545, 0.073386624, 0.060944743, 0.02807382]$$

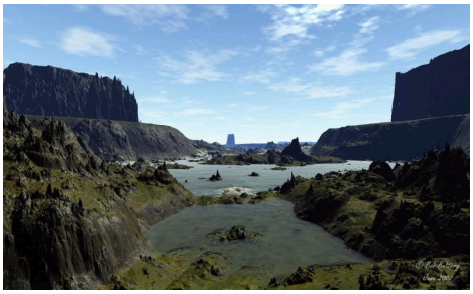
Classifier

We tried various classifiers available in python libraries such as Logistic Regression, Support Vector Machines(with various kernels). The training accuracy, as well as the testing accuracy were found to be quite close to each other, except in LDA, where accuracy was lower. In the final implementation, we have used Logistic Regression.

Observations

correctly identified photorealistic images:

incorrectly identified photorealistic images:




Accuracy

On using a dataset of size 3400 images, split as photographic and photorealistic, (1500 train, 200 test, each), procedure was repeated several times, each time with a new permutation. Following were the average accuracies obtained.

classifier	training accuracy	testing photographic	testing photorealistic
SVC	97.42	94.125	95.625
Logistic Regression	97.89%	95.75	96.25%

F1 score

classifier	F1 score
SVC	0.948
Logistic Regression	0.950

Other Observations

- Photographic images that contained reflecting surfaces were more difficult to classify correctly.
- Some computer generated images were realistic enough to be classified as photographic by our model.
- Obtained accuracy keeps increasing on increasing the size of dataset. Our model can do better, when more training images are provided. However, considering the capacities and time constraints, we have kept dataset size to 3400.

Conclusion

Analysing the distribution of wavelet coefficients can provide a lot of insight into the structure and origins of an image. As seen above, we can predict with about 96% accuracy that an image is photorealistic or not.

Extensions

The same procedure can be repeated for 1D signals, such as audio waves. In this case, we use 1D wavelet decomposition. The classification is between sounds recorded from a microphone and sound generated in a computer.

Motivation for second Technique

Due to recent advances in the field of artificial intelligence, algorithms based on deep neural networks (DNN) are more suitable for classification tasks. At the same time, when compared with existing methods, CNN-based algorithms can achieve real-time detection more effectively.

DNNs can extract image features efficiently and automatically train the classification features. In the field of image forensics, methods based on convolutional neural networks (CNNs) are the most popular implementations of DNNs and the most effective for solving these types of image forensics problems.

Considering that CNNs can describe image features with high performance, this paper will use CNNs to extract image features and train the recognition classification process.

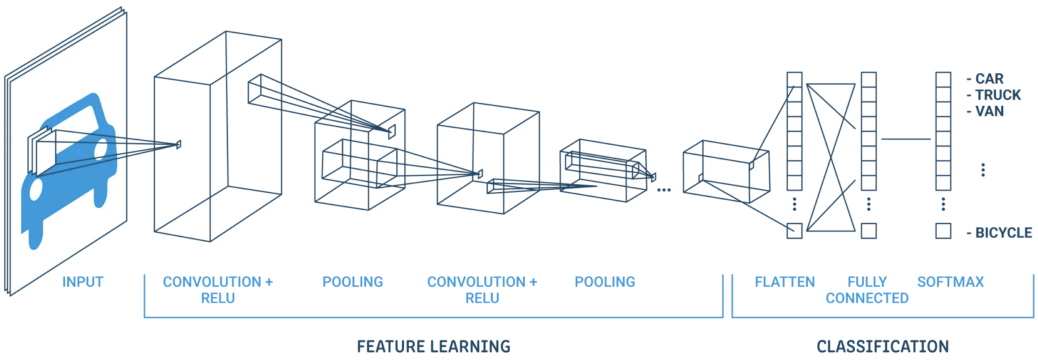


Figure 3: CNNs used for classification

CNNs are mainly used to identify image displacement, scaling, and other issues. Convolutional layers in the CNN structure utilize high-dimension vectors extracted from convolution kernels as image features. These high-dimension features are finally sent to the classification layer to complete the model training.

Approach

CNN

As a kind of deep neural network method, CNNs are widely used in the field of image processing.

The local perception strategy and parameter sharing strategy of CNNs can effectively reduce the number of parameters and improve training efficiency. Using the local network connection method separates out all but the closely-connected pixels in the image. By combining these local high-level features, the global features of the image can be obtained.

- The input image is decomposed by the convolutional layer into feature maps reflecting local dependency
- Pooling operations can reduce the association of feature maps to prevent overfitting, for example, as a common pooling operation, maximum pooling outputs the largest value of the selected area in the feature map, which allows for faster convergence
- Each convolution layer is activated by the relu function, which is expressed as:

$$relu(x) = max(x, 0)$$

- In the process of training, we choose the way of learning rate policy as step . Suppose the *base_lr*. The learning rate changes once after *stepsize* iterations. At the i-th iteration, the learning rate is:

$$lr = base\_lr \times \gamma^{i/stepsize}$$

- At the end of the network, the *softmax* connection will output the input samples to the probability output of each class. The network outputs the highest probability value as the prediction category. Assuming there are m samples, which are (X_i, Y_i) , in a total of L networks. The output of the last layer is$f(X_i)$. Then the loss function of the network is:

$$loss = -\frac{1}{m} \times Y_i log(f(X_i)) + \sum_{k=1}^{L} sum(\|W_k\|^2)$$

  where is the regularization parameter of weight W_k, Y_i is the output label of input X_i

High Pass Filter

Physical characteristics of the image, including CFA interpolation and pattern noise (mainly PRNU), have been shown to be effective for digital image forensics. Similar to the existing PRNU forensic methods, the input images are first subjected to highpass filtering during the training process.

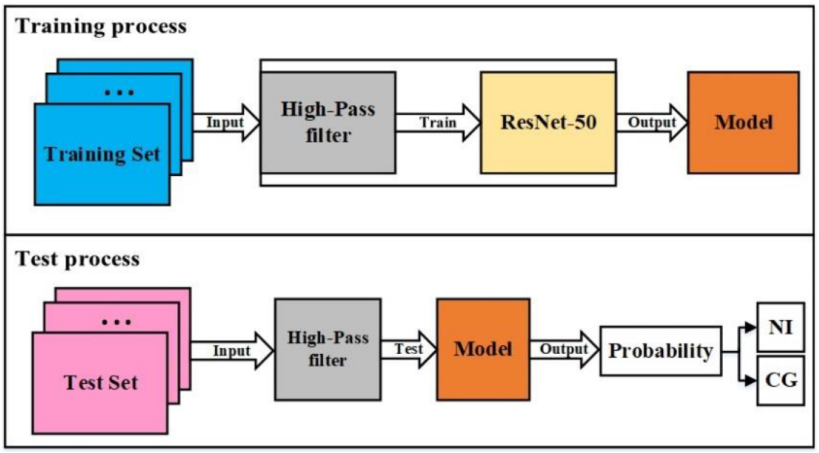


Figure 4: The proposed CNN structure for discriminating between PI and CG

Summary of Approach

1. In this model, each input image is grayed out first and then subjected to high-pass filtering to filter out the high-frequency components.
2. The noise is amplified by high-pass filtering, and the subsequent convolution layers of the network structure can use an optimization algorithm, such as gradient descent to fully learn the features of both types of images and use them to classify images across the two categories.

Experiments

Models

We have modified the CNNs network structure with a depth of 9 and 50 for experiments to prove that the deeper CNNs more suitable for this problem. In order to illustrate the effectiveness of the short-cut or skip connections for this problem, the accuracy and the loss of training of the same 9-layer network model are compared with and without skip connections.

Adam optimizer is used in all cases with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $weight\_decay = 0.0005$.

Model 1: A 9-layer CNN without skip connection
The CaffeNet architecture is used and shown on the left in Fig. 4. The network consists of 5 convolutional layers and 3 fully connected layers. Base learning rate in this case is 0.001 and is decreased after every 100 iterations to 10^{-7} at the end.

Model 2: The CaffNet used in Model 1 is modified to add shortcut connections. Base learning rate in this case is 0.001 and is decreased after every 100 iterations to 10^{-6} at the end.

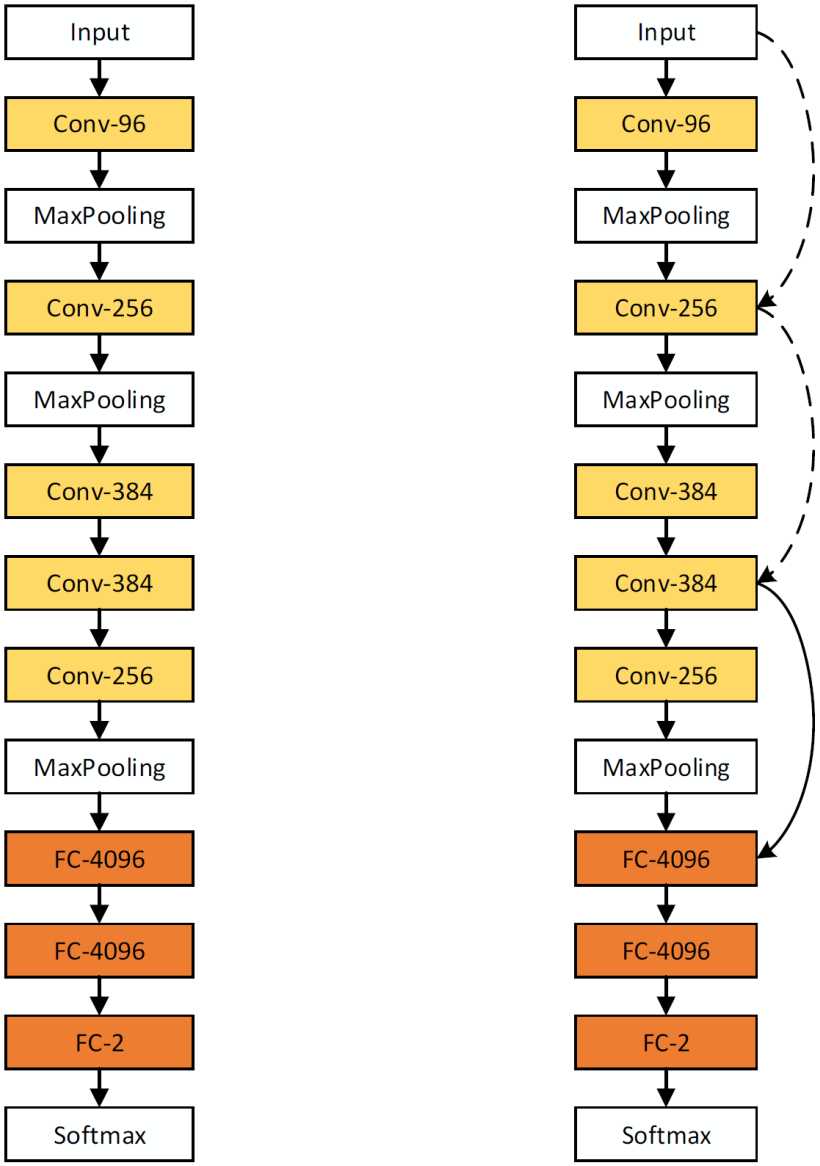


Figure 5: The left part is CaffeNet structure. The right part is CaffeNet with short-cut connection. The solid curve represents the direct short-cut connection and the dashed curve represents the modified short-cut connection.

Model 3: ResNet-50, a 50 layer CNN with skip connections is used. Base learning rate in this case is 0.0005 and is decreased after every 100 iterations to 5×10^{-8} at the end.

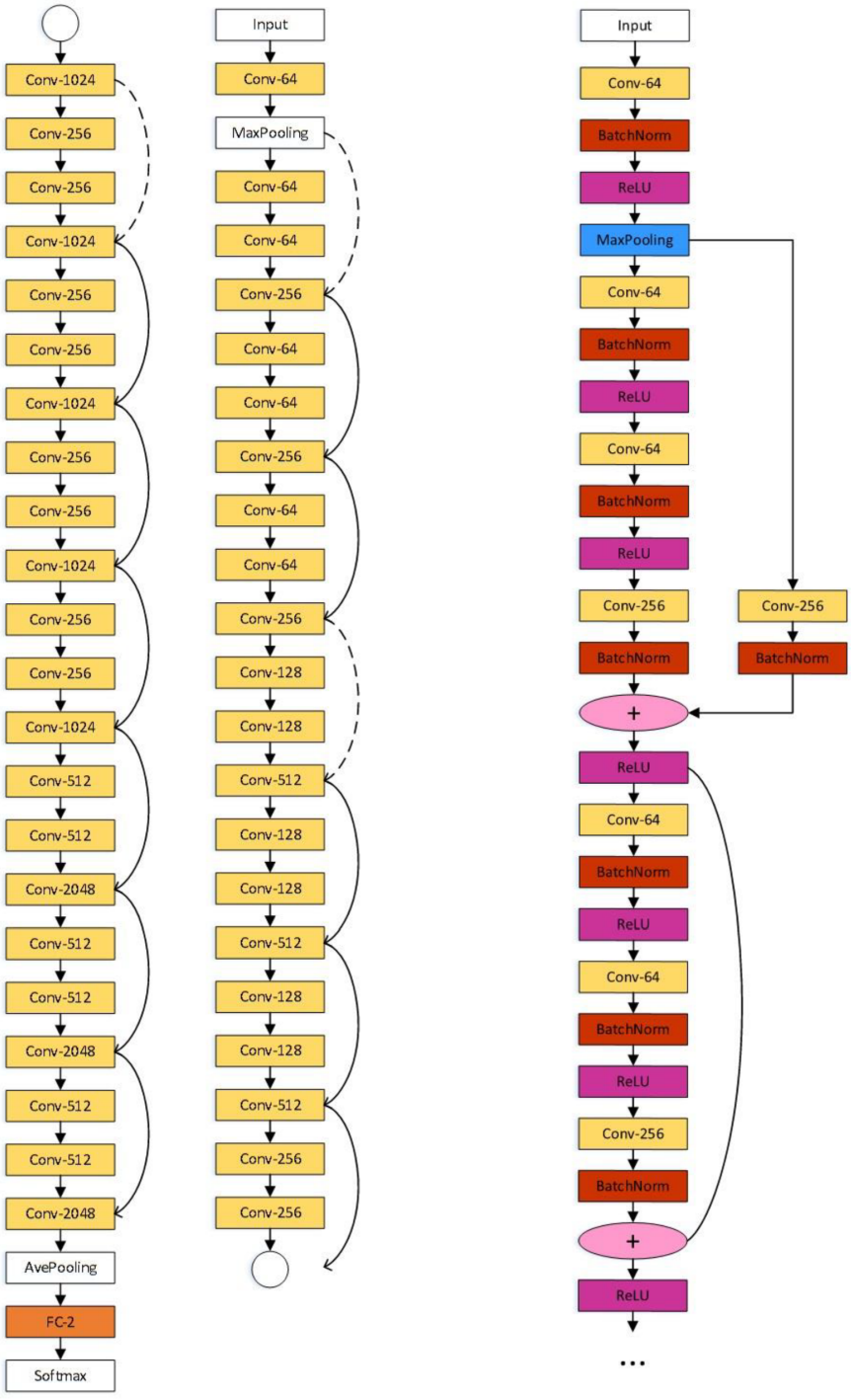


Figure 6: Part (a) is a 50-layer convolutional network with a short-cut connection. The solid curve represents the direct short-cut and the dashed curve represents the deformed short-cut. The white circles represent the connection symbol of the whole network.
Part (b) is an expanded view of the convolutions of the first seven layers on the left. The solid curve between the 7-layer convolutional structures indicates that there is a direct short-cut connection (solid curve on the left).
The plus sign indicates the element addition operation

Method of Filtering

We compared with 4 different methods of filtering and compared accuracy for each on training with Model-3.

800 photographic images and 800 photorealistic computer graphics are obtained for experiments from the Columbia Photographic Images and Photorealistic Computer Graphics Dataset.

Method 1: Using a 5×5 kernel and 2D filtering operation.

$$kernel = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 2 & 1 & -1 \\ -1 & 2 & 4 & 2 & -1 \\ -1 & 1 & 2 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

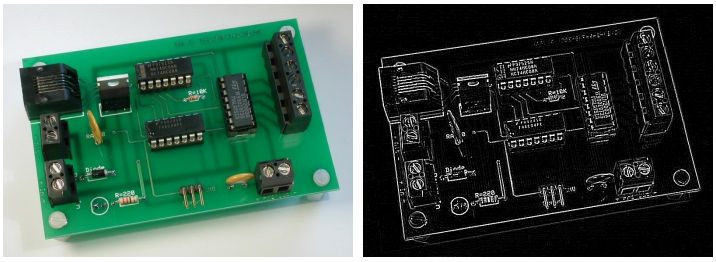


Figure 7: Original photographic image (left), Filtered image (right)

Figure 8: Original photorealistic image (left), Filtered image (right)

Method 2: Using the Median Blur filter from OpenCV. Median blur is a low pass filter, so subtracting it from the original image can give a high pass filtered version of image.

$$HPFimage = image - MedianBlur(image)$$

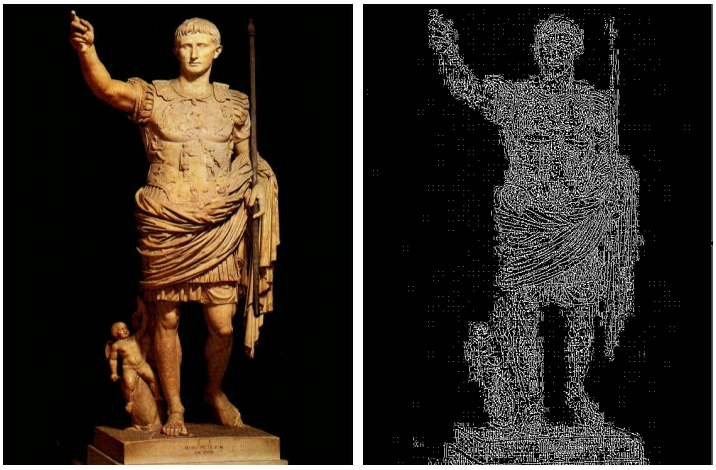


Figure 9: Original photographic image (left), Filtered image (right)

Figure 10: Original photorealistic image (left), Filtered image (right)

Method 3: Using the Gaussian Blur filter from OpenCV. Gaussian blur is a low pass filter, so subtracting it from the original image can give a high pass filtered version of image.

$$HPFimage = image - GaussianBlur(image)$$

Figure 11: Original photographic image (left), Filtered image (right)

Figure 12: Original photorealistic image (left), Filtered image (right)

Method 4: Frequency filtering done using Fast Fourier Transform of image.

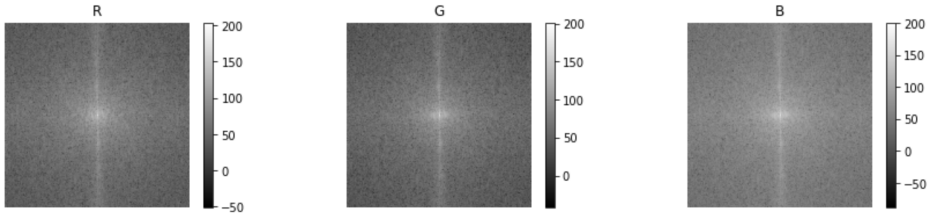


Figure 13: Original DFT for channel R, G and B

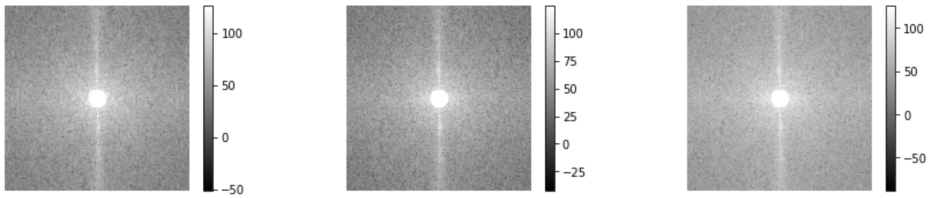


Figure 14: DFT after High Pass filtering

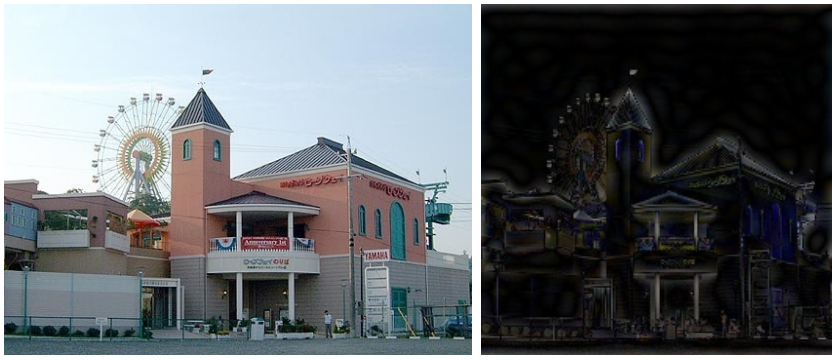


Figure 15: Original photographic image (left), Filtered image (right)

Figure 16: Original photorealistic image (left), Filtered image (right)

Visualization of Different HPFs

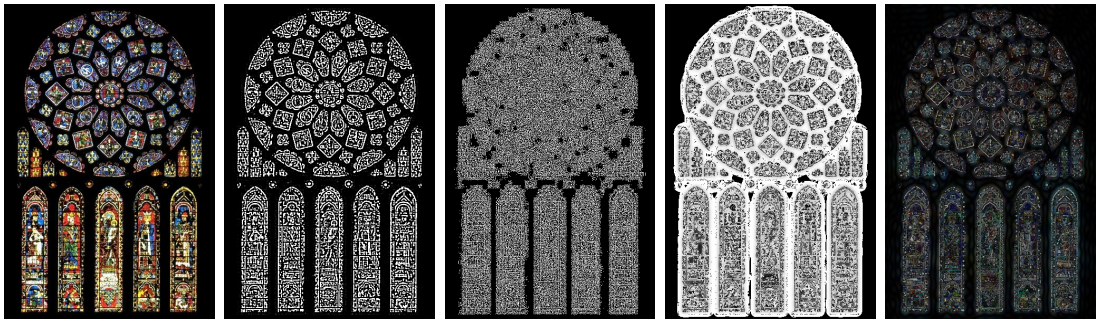


Figure 17: Original photographic image (first), HPF using 5×5 kernel (second), HPF by subtracting Median blur (third), HPF by subtracting Gaussian blur (fourth) and HPF using Fourier Transform (last)

Figure 18: Original photographic image (first), HPF using 5×5 kernel (second), HPF by subtracting Median blur (third), HPF by subtracting Gaussian blur (fourth) and HPF using Fourier Transform (last)

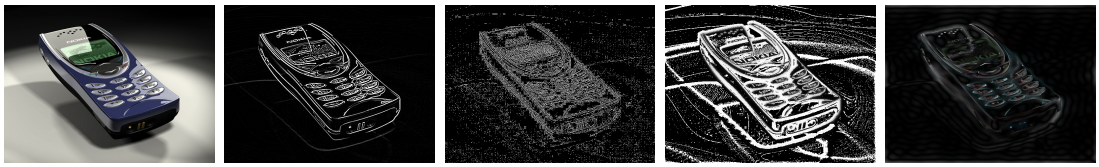


Figure 19: Original photorealistic computer graphic (first), HPF using 5×5 kernel (second), HPF by subtracting Median blur (third), HPF by subtracting Gaussian blur (fourth) and HPF using Fourier Transform (last)

Observations and Inferences

Training Process of CaffeNet

Gaussian Blurring is used to obtain high pass filter image and then passed to the model. After 1600 epochs, best training accuracy of 84.3% is achieved with test accuracy being 80.7%.

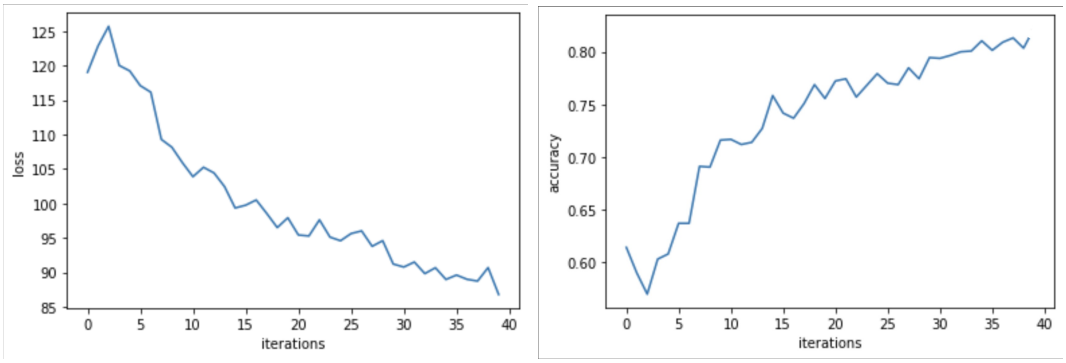


Figure 20: Training process of CaffeNet (iterations are scaled by 1/40)

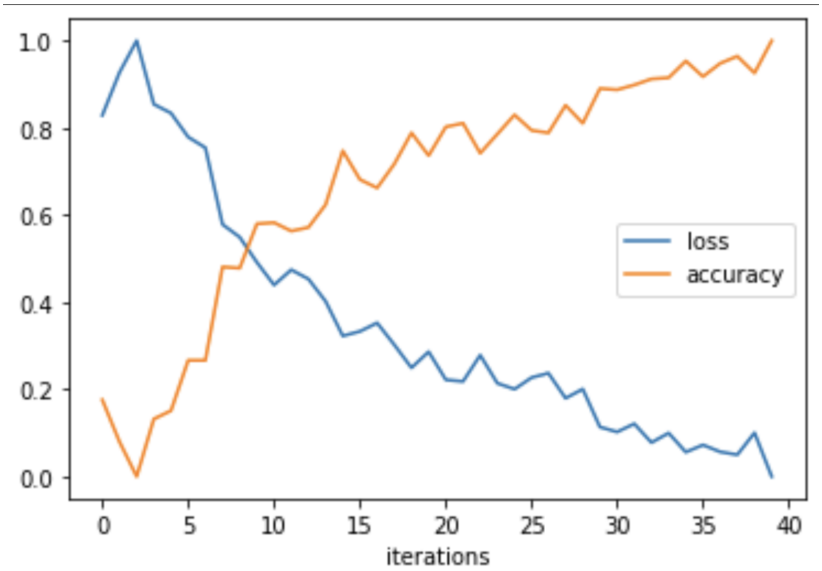


Figure 21: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

Training Process of modified CaffeNet

Gaussian Blurring is used to obtain high pass filter image and then passed to the model. After 3200 epochs, best training accuracy of 90.1% is achieved with test accuracy being 86.9%.

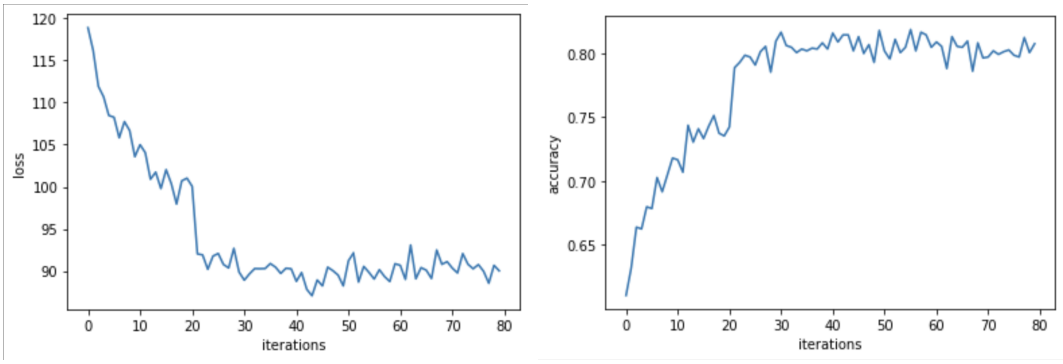


Figure 22: Training process of modified CaffeNet with skip connections (iterations are scaled by 1/40)

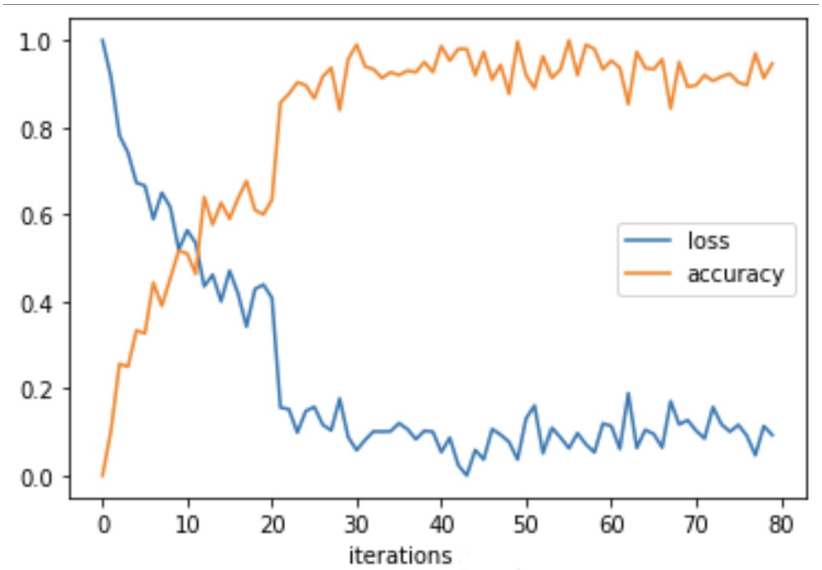


Figure 23: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

Training Process of ResNet-50

Four different experiments are conducted using the different types of high pass filters (HPF) described above. Their observations are -

HPF using a 5×5 kernel

After 2400 epochs, best training accuracy of 91.8% is achieved with test accuracy being 90.7%.

HPF using Median Blur

After 2400 epochs, best training accuracy of 95.8% is achieved with test accuracy being 93.0%.

HPF using Fast Fourier Transform

After 2400 epochs, best training accuracy of 94.1% is achieved with test accuracy being 91.4%.

HPF using Gaussian Blur

After 2000 epochs, best training accuracy of 98.1% is achieved with test accuracy being 95.6%.

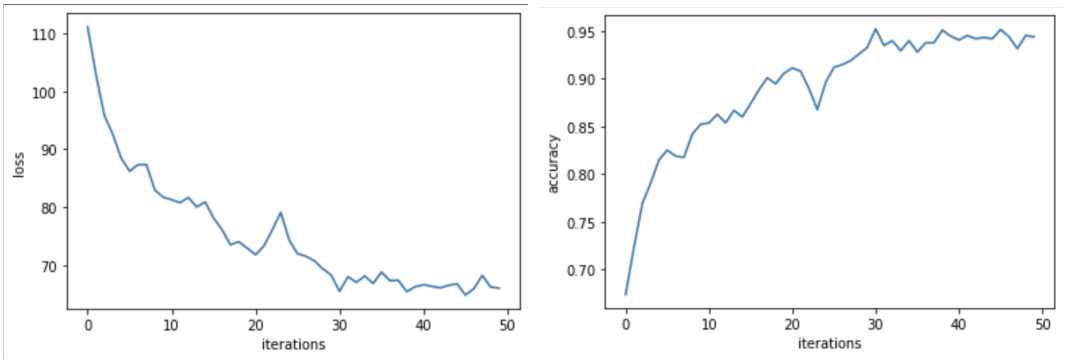


Figure 24: Training process of ResNet-50 with HPF obatined using Gaussian blurring (iterations are scaled by 1/40)

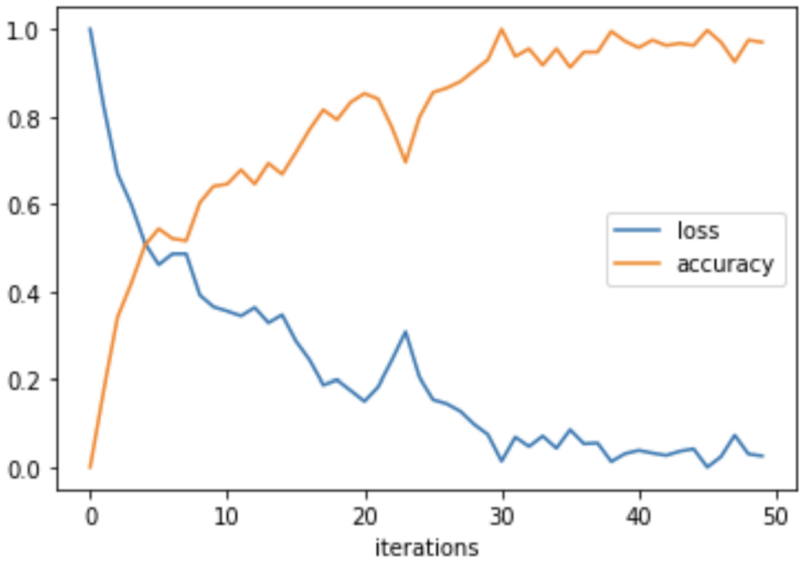


Figure 25: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

Comparison and Analysis

Accuracy of different models when using Gaussian blurring for obtaining the high pass filter -

Model	Training Accuracy	Test Accuracy
CaffeNet	84.3%	80.7%
Modified CaffeNet	90.1%	86.9%
ResNet-50	98.1%	95.6%

Accuracy of ResNet-50 with different techniques for obtaining high pass filter (HPF) -

HPF type	Training Accuracy	Test Accuracy
5×5 kernel	91.8%	90.7%
Median blur	95.8%	93.0%
Gaussian blur	98.1%	95.6%
Fourier Transform	94.1%	91.4%

Comparison of network with or without short-cut connection

We observe that after increasing the short-cut connection, the detection accuracy of the network is improved from 80.7% to 86.9%, and the loss function during the training process drops more quickly, indicating that the short-cut connection can solve the gradient vanishing problem. At the same time, the network model will pay more attention to the important classification features of input images.

Thus, adding shortcut connections or *skip connections improves accuracy.*

Comparison of the depth of the network

We observe that after increasing the depth of the network, the detection accuracy can be increased to more than 95%, which indicates that deepening the network layers can *effectively improve the classification accuracy.*

Comparison of different High Pass Filters

We observe that the high pass filter obtained by subtracting Gaussian blur of image of from original image can outperforms all others on ResNet-50. Thus, for the purpose of differentiating between photographic images and photorealistic computer graphics, the image obtained after subtracting its Gaussian blurring retains the most important features and information.

Predictions of the Best Model

We consider ResNet-50 to be our best model when High Pass Filter is obtained by subtracting the Gaussian blur of image from original image.

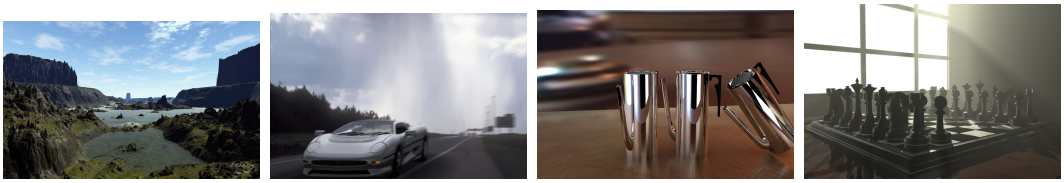


Figure 26: Some photorealistic computer graphics from the Columbia dataset which were classified correctly by our best model

Figure 27: Some photographic images from the Columbia dataset which were classified correctly by our best model

Results of Extension

As our 50-layer Resnet with High Pass Filter achieves very high accuracy on classifying Photographic Images and Computer Graphics, it had potential to be used in other domains of image forensics. In particularly, classifying AI generated images from photographic images and computer graphics.

AI Generated Images

Generative Adversarial Networks, or GANs for short, are an approach to generative modeling using deep learning methods, such as convolutional neural networks. Using GANs, AI models are capable of generating fake faces that look almost identical to real people. They have become really good at it over the years, and it is almost impossible to distinguish between a real face and an AI generated fake.

The images shown below have been downloaded from a website `thispersondoesnotexist.com` which generates fake images of human faces using a GAN. As we can see, these images are very relaistic and very difficult to tell part from photographic images or computer graphics.

GAN-generated images can be very convincing. This can be dangerous, since GANs

can be used to create fake dating profiles, catfish people, and spread fake information. It is very important for us to be able to distinguish between fake and real.

Dataset

The style-based GAN architecture (StyleGAN) yields state-of-the-art results in data-driven unconditional generative image modeling. Nvidia made their models StyleGAN and StyleGAN2 open source and also published a list curated images generated by their model. We acquired 800 of those images for training our classifier.

Figure 28: The GAN generated images acquired belonged broadly to the same categories as the images in our dataset of photographic images and computer graphics

Experiments and Results

GAN generated images vs Photographic images

For obtaining a high pass filter, we subtracted the Gaussian blurred version of image from original image and then passed it into the classifier ResNet-50. Base learning rate was set at 0.0001 and decreased by a factor of 0.1 during training in case accuracy did not improve for 4 successive iterations.

Adam Optimizer was used with hyperparameter values as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used for optimization.

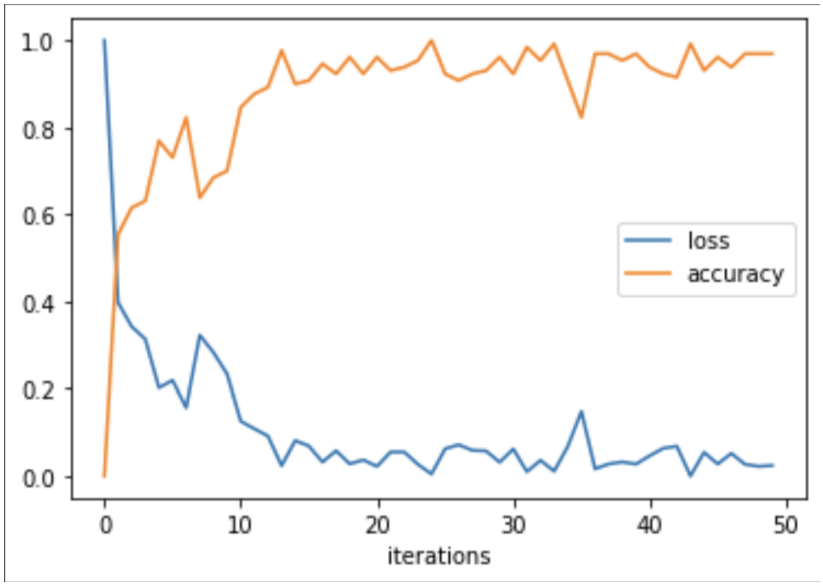


Figure 29: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

After only 2000 iterations, our model achieved a test set accuracy of 98.2% and the best training accuracy was 99.1%.

GAN generated images vs Computer Graphics

For obtaining a high pass filter, we subtracted the Gaussian blurred version of image from original image and then passed it into the classifier ResNet-50. Base learning rate was set at 0.0005 and decreased by a factor of 0.1 during training in case accuracy did not improve for 4 successive iterations.

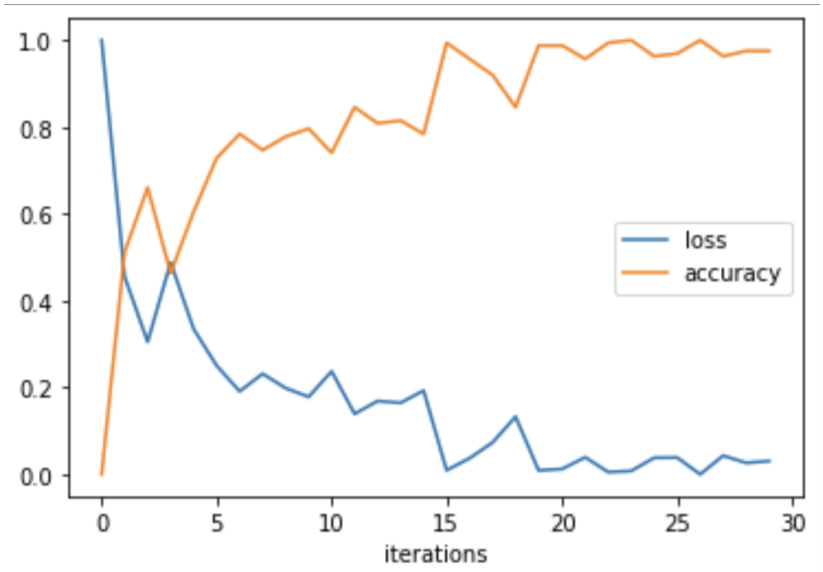


Figure 30: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

Adam Optimizer was used with hyperparameter values as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used for optimization again.

After only 1200 iterations, our model achieved a test set accuracy of 96.4% and the best training accuracy was 98.2%.

GAN generated images vs Photographic images vs Computer Graphics

For obtaining a high pass filter, we subtracted the Gaussian blurred version of image from original image and then passed it into the classifier ResNet-50. Base learning rate was set at 0.0001 and decreased by a factor of 0.1 during training in case accuracy did not improve for 4 successive iterations.

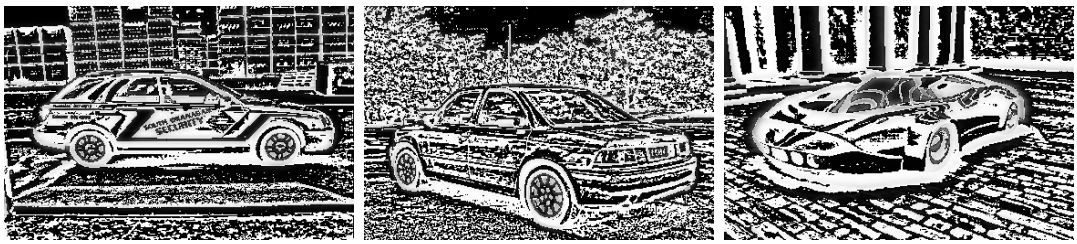


Figure 31: High pass filtered versions of (a) photographic image (b) GAN generated image (c) computer graphics. (b) has much highest irregularity while (c) is the smoothest. (a) is midway between the two.

Adam Optimizer was used with hyperparameter values as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used again.

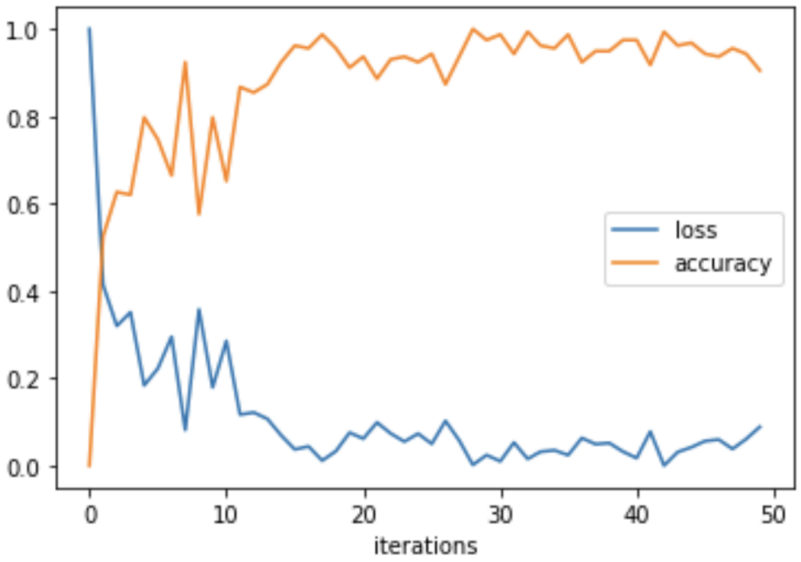


Figure 32: Scaled values of loss and accuracy during the process of training (iterations are scaled by 1/40)

After 2000 iterations, our model achieved a test set accuracy of 91.2% and the best training accuracy was 95.1%.

Model Predictions

The following images were predicted accurately by our model.

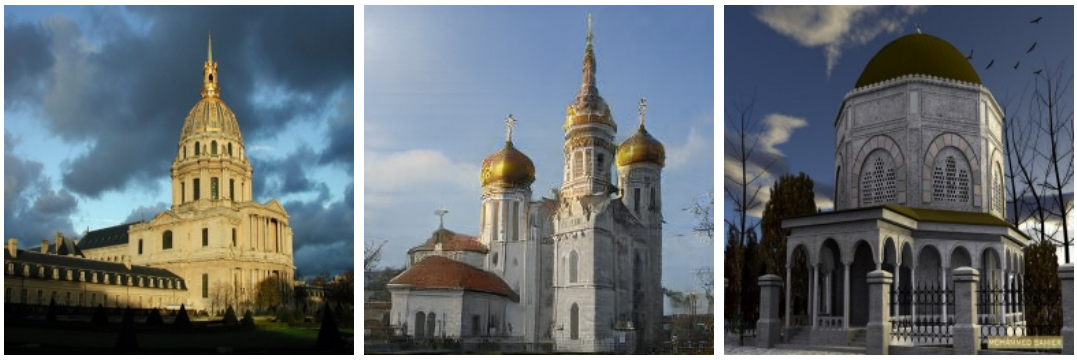


Figure 33: (a) photographic image (b) GAN generated image (c) computer graphics

Figure 34: (a) photographic image (b) GAN generated image (c) computer graphics

Conclusion and Future Work

Our High Pass Filter Filtering technique accompanied with our CNN model has very high accuracy in distinguishing GAN generated images from photographic images. Since a pre-trained model can produce results almost instantly, it can very useful in real-time image forensics especially on social media.

References for Datasets and sofware used

PyTorch: Paszke, A. et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035.
Available at: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

OpenCV: Bradski, G., 2000. The OpenCV Library. Dr. Dobbx27;s Journal of Software Tools

Scipy: Virtanen, P. et al., 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, pp.261–272

Scikit-learn: Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825–2830

Matplotlib: Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Computing in science amp; engineering, 9(3), pp.90–95

Numpy: Harris, C.R. et al., 2020. Array programming with NumPy. Nature, 585, pp.357–362

PIL: Clark, A., 2015. Pillow (PIL Fork) Documentation, readthedocs.
Available at: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

PI and CG dataset: Ng et al., 2005. Columbia photographic images and photorealistic computer graphics dataset. Columbia University, ADVENT Technical Report, pp.1-23

GAN image dataset: T. Karras et al., 2020. Analyzing and Improving the Image Quality of StyleGAN. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8107-8116

ResNet: Kaiming H. et al., 2016. Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778

GAN image dataset: T. Karras et al., 2020. Analyzing and Improving the Image Quality of StyleGAN. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8107-8116

GAN image dataset: T. Karras et al., 2020. Analyzing and Improving the Image Quality of StyleGAN. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8107-8116