Seam Carving for Image Classification Privacy

James Pope[a] and Mark Terwilliger
Department of Computer Science and Information Systems,
University of North Alabama, Florence, Alabama, U.S.A.

Keywords: Privacy Protection, Adversarial Perturbations, Image Classification, Seam Carving.

Abstract: The advent of storing images on cloud platforms has introduced serious privacy concerns. The images are routinely scanned by machine learning algorithms to determine the contents. Usually the scanning is for marketing purposes but more malevolent purposes include criminal activity and government surveillance. The images are automatically analysed by machine learning algorithms. Notably, deep convolutional neural networks perform very well at identifying image classes. Obviously, the images could be encrypted before storing to cloud platforms and then decrypted after downloading. This would certainly obfuscate the images. However, many users prefer to be able to peruse the images on the cloud platform. This creates a difficult problem in which users prefer images stored in a way so that a human can understand them but machine learning algorithms cannot. This paper proposes a novel technique, termed *seam doppelganger*, for formatting images using seam carving to identify seams for replacement. The approach degrades typical image classification performance in order to provide privacy while leaving the image human-understandable. Furthermore, the technique can be largely reversed providing a reasonable facsimile of the original image. Using the ImageNet database for birds, we show how the approach degrades a state-of-the-art residual network (ResNet50) for various amounts of seam replacements.

1 INTRODUCTION

With the advent of image recognition machine learning approaches, numerous privacy issues have been raised. More recently, deep learning techniques, such as deep layered convolutional neural networks (CNNs), have made significant advances in image (Traore et al., 2018) and speech recognition (Kumar et al., 2018) compared to traditional machine learning approaches, exacerbating the privacy concerns. Recent research has been to provide privacy-preserving image techniques to counter these classifiers (Moosavi-Dezfooli et al., 2016). For example, researchers (Sanchez-Matilla et al., 2020) have suggested exploiting vulnerabilities in deep neural networks to protect image privacy. Conversely, researchers have investigated image compression techniques to counter these adversarial images (Liu et al., 2019) (Das et al., 2017).
In general, machine learning approaches attempt to learn correlations to predict the target class. Redundant information is ideal when noise is considered and generally improves the performance of many classifiers. Given that general machine learning approaches exploit redundant information to improve accuracy, an obvious counter would be to minimise redundant information in images as much as possible while still retaining enough for humans to recognise the image. Of course, human recognition is very subjective.
Seam carving was proposed by Avidan and Shamir (Avidan and Shamir, 2007) for content-aware image resizing for both reduction (i.e. compression) and expansion. Seam carving can be used to compress images by finding the least informative pixels (determined to be redundant) in a corresponding energy image and discarding them along contiguous vertical and horizontal paths. We propose modifying seam carving to replace these less informative pixels using a deterministic function of nearby pixels that will remove the redundancy while keeping the image human-perceptible.
To our knowledge, our work is the first to leverage seam carving for privacy preserving images. The contributions of this paper are the following.

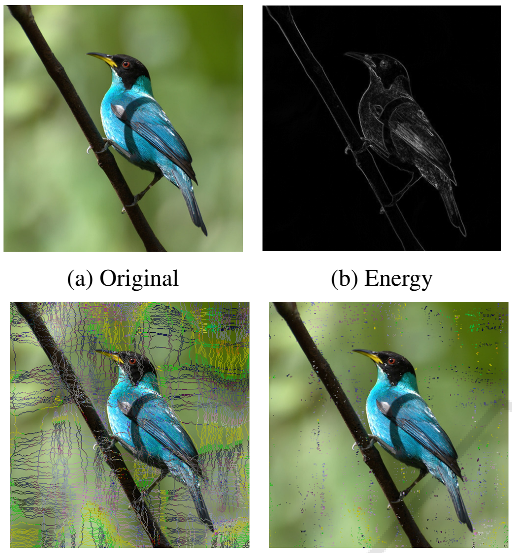
- Novel approach for privacy preserving images by modifying the seam carving image compression technique

[a] https://orcid.org/0000-0003-2656-363X

Pope, J. and Terwilliger, M.
Seam Carving for Image Classification Privacy.
DOI: 10.5220/0010249702680274
In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2021)*, pages 268-274
ISBN: 978-989-758-486-2

- Comparison with deep CNNs showing the efficacy of our approach

We also submit that this work can be reproduced providing sufficient information to produce similar results as the images are available from ImageNet and the code is made freely available (Pope, 2020).


(a) Original (b) Energy

(c) Replaced (d) Restored

Figure 1: Seam Doppelganger Example: Jacamar.

This paper first details the seam doppelganger approach including seam replacement and seam restoration. The approach is then evaluated using specific images from a common image repository and a modern image classifier. An experiment on several hundred images is then conducted, comparing the seam doppelganger approach to random distortions. The paper closes with future work and the conclusion.

2 SEAM DOPPELGANGER APPROACH

The seam doppelganger approach includes two steps, seam replacement and seam restoration. The seam carving technique is first briefly explained followed by seam replacement and then restoration.

2.1 Seam Carving and Energy Function

The seam carving approach (Avidan and Shamir, 2007) uses an energy function to transform the red, green, and blue (RGB) components of the original image into an energy image. Energy functions can measure the energy at a pixel in various ways and typically include surrounding pixels as inputs. For our research, we use a common differential energy function that takes the difference of the left and right pixel colours and adds to the difference of the top and bottom pixel colours. The difference between two colours is determined by Equation 1 and the energy at the pixel location's i'th row and j'th column is given by Equation 2.

$$\begin{aligned} diff(c1,c2) = (c1.red - c2.red)^2 + \\ (c1.green - c2.green)^2 + \\ (c1.blue - c2.blue)^2 \end{aligned} \tag{1}$$

$$\begin{aligned} c_w &= image(i-1,j) \\ c_e &= image(i+1,j) \\ c_n &= image(i,j-1) \\ c_s &= image(i,j+1) \\ e(i,j) &= diff((c_w,c_e) + diff(c_n,c_s) \end{aligned} \tag{2}$$

This is similar to an edge detector kernel and records higher energy at boundaries between colour regions. In areas with the same or similar colour, a lower energy value is computed. Note that the values computed in Equation 1 and 2 are not normalised. By normalised, we mean to keep in the range of $[0,255]$. We submit that this unnormalised energy function is sufficient for computing shortest paths. Normalised values are simply re-scaled and the same paths will be found.

A horizontal seam is determined from the energy image by finding a constrained shortest path from the left to the right of the image. The horizontal seam length is constrained to be the width of the image. The seam carving authors refer to the seam as an 8-connected path that can be found in a variety of ways, including Dijkstra's shortest path algorithm or dynamic programming (Avidan and Shamir, 2007). Our implementation uses a variation of the shortest path algorithm. Similarly, the vertical seam is the shortest path in the energy image from top to bottom with a length of the image height.

Figure 1a shows an original image and Figure 1b show the corresponding energy image produced using the normalised energy function (slightly enhanced to make brighter for presentation). Note that in Figure 1c, seam carving would either remove these seams (for compression) or expand them. Instead we replace the seams.

2.2 Seam Replacement

For each horizontal and vertical seam found, seam carving for compression would omit it reducing the

size of the image. The seam is considered the most redundant and therefore best to remove without loss of information. For our purposes, we choose to replace the seam leaving the image the exact same size but with the seam's pixels replaced.
Each seam found and replaced produces a new image. Should we want to replace more than one seam, we have to run the entire procedure again including producing a new energy image.

Algorithm 1: Seam Replacement (Horizontal).

```
 1: Input: image $X$ and $n$ number of seams to replace
 2: Output: image $Y$ copy of $X$ with $n$ seams replaced
 3: **procedure** HORIZONTALREPLACEMENT($X, n$)
 4:     $Y = \text{copy}(X)$
 5:     **while** seams replaced $< n$ **do**
 6:         seam = findHorizontalSeam(Y)
 7:         replaceHorizontalSeam(Y, seam)
 8:     **return** $Y$
 9:
10: **procedure** FINDHORIZONTALSEAM($Y$)
11:     $E = \text{energyImage}(Y)$
12:     **for** each row $i$ in E.height **do**
13:         path = findShortestPath(E,i)
14:         **if** length(path) $<$ smallest path so far **then**
15:             smallestSeam = path
16:     **return** smallestSeam
17:
18: **procedure** REPLACEHORIZONTALSEAM($Y, seam$)
19:     **for** each col $j$ in Y.width **do**
20:         i = seam(j)
21:         // Note we ignore value at $i, j$ and set as
22:         // a function of east, west pixel values
23:         maskValue = mask(i, j, Y)
24:         Y.set(i,j, maskValue)
25:
26: **procedure** MASK($i, j, Y$)
27:     north = Y.get(i,j-1)
28:     south = Y.get(i,j+1)
29:     rn = north.red
30:     gn = north.green
31:     bn = north.blue
32:     rs = south.red
33:     gs = south.green
34:     bs = south.blue
35:     r = ( bn + gs + j ) & 0xC5
36:     g = ( rn + bs + j ) & 0xC5
37:     b = ( gn + rs + j ) & 0xC5
38:     **return** Colour(r, g, b)
```

Algorithm 1 shows the procedures for replacing horizontal seams (vertical seams are replaced in a similar fashion). The **HorizontalReplacement** (Line 3) and **findHorizontalSeam** (Line 10) procedures are similar to the original seam carving approach. We differ at Line 7 where we replace instead of elide or expand the seam. The **replaceHorizontalSeam** (Line 18) procedure enumerates each pixel location in the seam and replaces with a mask colour value derived from the surrounding pixels. The **mask** procedure (Line 26) details how this colour is determined.
There are two desirable properties regarding seam replacement. First, we would like for subsequent seams to not find the replaced seam (i.e. the seam should have relatively high energy values). The second property is that we would like to be able to find the seam later so that we can attempt to restore the values.
The **mask** procedure first extracts the red, green, and blue colour components from the pixels to the north and south of the pixel to be replaced. At Lines 35, 36, and 37, each colour component is mixed with a combination of the north and south, however, using different colour components. For instance at Line 35, red is mixed with blue from the north pixel and green from the south pixel. The intent of this mixing is to reduce the possibility of the transformed colour being similar (i.e. to make it stand out from the east and west colour). If the derived colour is very similar then it is redundant and will likely be found again as part of the next seam to replace. Furthermore, the column location j is also mixed with each colour component. If there is a series of similar colour values to the north and south, adding the column location causes the mask colour to change gradually giving a gradient effect. We believe that this will better confuse an image classifier as the replaced seams will be correlated creating their own patterns. Finally the sum is a bit-wise *and* with $0xC5$. This takes the lower eight bits in a hash-like way to ensure the resulting value is in the range of $[0, 255]$. The $0xC5$ is for convenience and other approaches may work equally well.
We submit that the **replaceHorizontalSeam** procedure with the mask colours achieves the two desirable properties. The colours are sufficiently different from neighbouring pixels to avoid re-selection of the seam pixels for replacement. Second, the mask colour values are computed using a deterministic function of the neighbouring pixels (NB: as long as the neighbouring pixel values do not change later).

2.3 Seam Restoration

Based on the seam replacement, the original picture cannot be 100% restored perfectly. The seam values are completely replaced as a function of the neighbouring pixel values. However, the seam pixels by definition are highly redundant. Thus, we can sim-

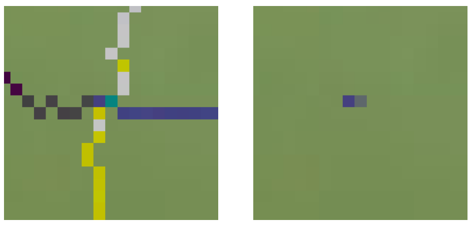

(a) Region 20x20 of original image with horizontal and vertical seams replaced (b) Region 20x20 seam restoration with artefacts from horizontal seam

Figure 2: Seam Restoration Example.

ply take the pixel value to be the average of the north and south pixels for horizontal seams (and west and east for vertical seams). To detect a replaced pixel we check to see if its colour is equal to the value computed by the mask procedure. We check to see if it could have been a pixel in a vertical seam and then if it could have been a pixel in a horizontal seam. We check every pixel in this way. To be clear, we do not explicitly find the seam again. If the pixel could have been replaced, we assume it was and take the average of the neighbouring pixel values overwriting its current colour value.
A notable exception to replaced pixels is when we first replace horizontal seams followed by replacing vertical seams. Horizontal seams have to intersect with vertical seams. The vertical replacement will overwrite the neighbouring pixels used to mask the horizontal seam and it will not be able to detect that it was replaced. Though the vertical seam's masked value will be found, the horizontal seam's pixel will not be detected, leaving artefacts. Figure 2a shows the results of a 20x20 pixel region (near the back of the bird) of the image in Figure 1c where a horizontal and then vertical seam were replaced. Figure 2b shows the corresponding results of the seam restoration. A masked horizontal pixel near the centre was not detected and remains as an objectionable artefact (darker blue). Future work is to address this issue. Note that the vertical seam was replaced and the dark green pixel near the centre was restored as the average of the left and right colours (this shows as a lighter blue pixel next to the darker blue artefact).
Figure 2 clearly shows that the horizontal and vertical seams can largely be restored with minimal distortion of the original image. This is because, by design, the pixels replaced were redundant and not very informative. Restoring then from neighbouring pixels results in a near facsimile of the original image.

Table 1: ImageNet Query.

Class	# Instances	Search
bird	608	Animal, animate being, beast, brute, ... Chordate Vertebrate, craniate Bird

3 EVALUATION

This section evaluates the seam doppelganger approach by applying it to selected images from a common image repository and examining how it affects an image classifier. The evaluation shows that many replaced images are still human perceptible but degrade the classifier's performance. We also show a counter example where the approach fails.

3.1 Experimental Data

The images are retrieved from ImageNet (Russakovsky et al., 2014). To make the analysis more coherent, we narrowed to just the birds synsets. The images for each were queried from the ImageNet repository where the image URLs were saved to a file. The images were then retrieved restricting to just the URLs starting with *http://farm*. The ImageNet query (IMA, ease) and resulting number of instances are shown in Table 1.
The images were resized to be 244 x 244 (similar to VGG (Simonyan and Zisserman, 2014)) with the three red, green, and blue channels. The values were normalised to be between 0.0 and 1.0.
We would like to show that our seam doppelganger approach works comparable to randomly modifying the image. To ensure comparisons are valid, the same number of pixels are modified for both the random and seam doppelganger approaches. The following equation is used to calculate the number of pixels modified for the seam approach.

$$p = r \times image.width + c \times image.height - (r \times c)$$

where p is the number of pixels modified, r is the number of rows seamed, and c is the number of columns seamed.
Note that we are careful to subtract the row-column intersections as they are modified twice.

3.2 Albatross

Figure 3 shows the four variations of the image. Subfigure 3a shows the original image from ImageNet

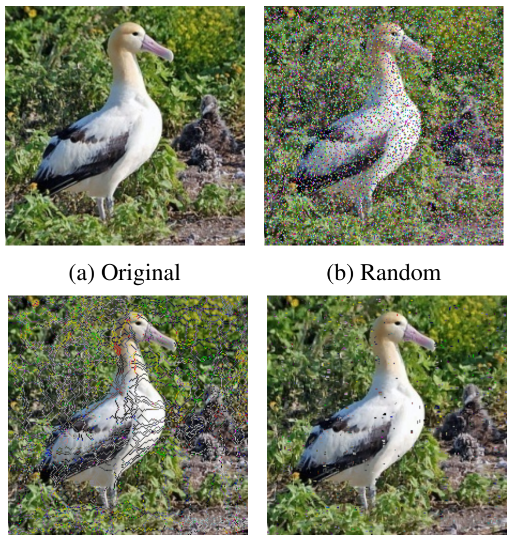


Figure 3: Examples Images: Albatross 30% Replacement.

Table 2: Example Prediction Comparison *Albatross*.

Scenario	Class	Probability
Original	albatross	0.974
	pelican	0.006
	stork	0.006
Random	ptarmigan	0.383
	albatross	0.313
	drake	0.160
Replaced	ptarmigan	0.848
	bustard	0.065
	albatross	0.044

(ALBATROSS_027) that is 224 pixels wide by 224 pixels tall. Sub-figure 3c shows where 34 row seams and $\times$ 34 column seams have been replaced. This replaces a total of 14076 pixels, roughly 30% of the image (34 is roughly 15% of the width and height). Sub-figure 3b shows the same number of pixels modified by randomly selecting pixels and then changing them to a random colour. To avoid modifying the same pixel twice, we assign a unique integer to each pixel location, shuffle these indices, and then select the first p to modify. Finally, sub-figure 3d shows the results of the restoration process run on sub-figure 3c. There are clear artefacts where the seams cross and also where two parallel seams were close and overwrote each other (e.g. back of the bird's neck). However, the image well represents the original.
We use ResNet50 to predict the original, random, and replaced images. We take the *top-3* predicted classes and show their probabilities in Table 2.

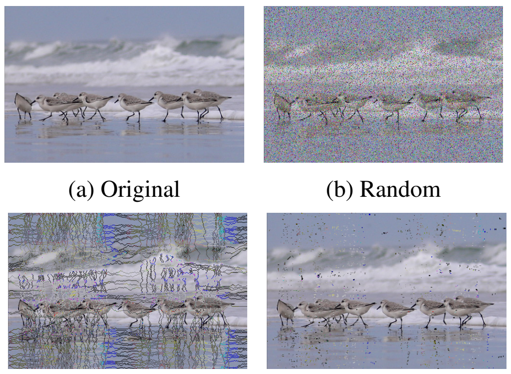


Figure 4: Examples Images: Dowitcher 30% Replacement.

Table 3: Example Prediction Comparison *Dowitcher*.

Scenario	Class	Probability
Original	dowitcher	0.827
	red-backed_sandpiper	0.154
	ruddy_turnstone	0.014
Random	chainlink_fence	0.233
	ant	0.222
	window_screen	0.123
Replaced	padlock	0.320
	window_screen	0.181
	jigsaw_puzzle	0.161

3.3 Dowitcher

Figure 4 shows several dowicher birds on the beach with a 30% random and seam replacement. Table 3 shows the top-3 results of running ResNet50 against the original, random, and replaced images.

3.4 Goose

Figure 5 shows a single goose with 30% of the pixels replaced randomly and by seam replacement. Table 4 shows the top-3 results of running ResNet50 against the original, random, and replaced images.

Table 4: Example Prediction Comparison *Goose*.

Scenario	Class	Probability
Original	goose	0.496
	flamingo	0.394
	albatross	0.045
Random	ostrich	0.736
	eel	0.200
	starfish	0.013
Replaced	ostrich	0.968
	flamingo	0.022
	crane	0.003

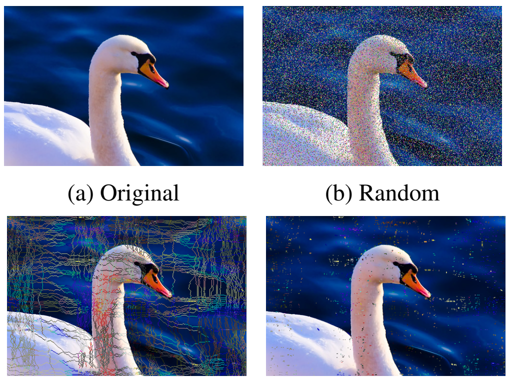

(a) Original (b) Random
(c) Replaced (d) Restored

Figure 5: Examples Images: Goose 30% Replacement.

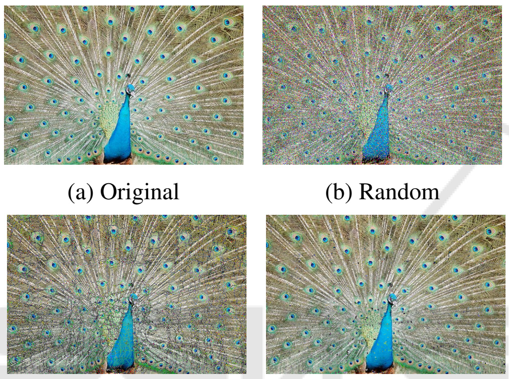

(a) Original (b) Random
(c) Replaced (d) Restored

Figure 6: Examples Images: Peacock 30% Replacement.

In the original image, the classifier is already confused between *goose* and *flamingo*. Interestingly in this case, the certainty of the classifier is increased for both the random and replaced images, however, with a class that is not in the top three of the original image.

3.5 Peacock

Figure 6 shows a peacock with 30% of the pixels replaced randomly and by seam replacement. Table 5 shows the top-3 results of running ResNet50 against the original, random, and replaced images.
This counter example shows that the seam replacement approach does not always work. The peacock image does not have easily identified redundancy and the seam replaced image is not noticeably different than the original. The classifier performance is hardly degraded at 0.997 certain that the image is a peacock.

Table 5: Example Prediction Comparison *Peacock*.

Scenario	Class	Probability
Original	peacock	0.999
	fountain	0.000
	doormat	0.000
Random	chainlink_fence	0.593
	chain	0.208
	chain_mail	0.121
Replaced	peacock	0.997
	chainlink_fence	0.000
	shovel	0.000

4 EXPERIMENT AND RESULTS

To demonstrate the effectiveness of the approach, the predictions of Resnet50 are compared to random and seam replaced images with varying amounts of replacement (a.k.a. distortion). We use the average prediction probability for measurement comparison.

4.1 Average Prediction Probability

Based on the 608 bird images from ImageNet, we seam replace each image with 10%, 20%, 30%, 40%, and 50% of the pixels replaced. For comparison, we also change exactly this same number of pixels randomly. Resnet50 is run on the original images (0% distortion), the seam replaced images, and the randomly distorted images. As shown previously, some image probabilities will be worse and some may be better. We take the average of the image predictions for each distortion percentage. Ideally, increasing the distortion will cause the average probability to decrease for both the seam replaced and randomly distorted images. As a comparison, the random results are treated as an ideal distortion of the images for our privacy preserving scenario. We expect the random distortions to reduce more than the seam replacement. However, it is desirable that the seam replacement results be close to the random results.

Figure 7 shows the results of the experiment. The results for both the seam replaced and randomly distorted images does indeed decrease as the distortion increases. Importantly, the seam replaced results remain close to the random results for each amount of distortion. These results provide evidence that the seam carving approach can be used to degrade the prediction probability of deep neural network image classifiers.

At 30% distortion, the probability prediction of Resnet50 on the random images continues to reduce closer to zero. In actuality, each of the 1000 image

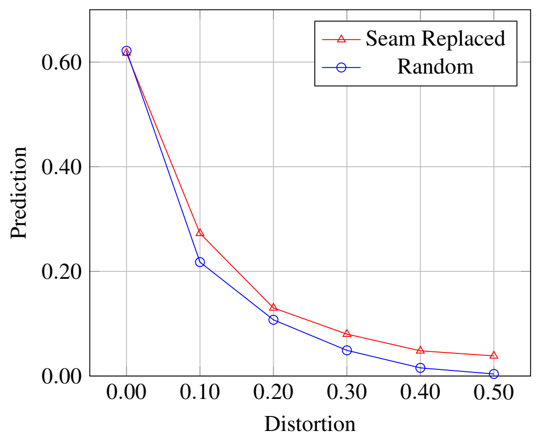


Figure 7: Resnet50 predictions on 608 distorted bird images.

classes is becoming equally likely (i.e. maximum entropy). The seam carving approach is levelling out at about 3-4% for the 50% distorted images.

4.2 Experimental Issues and Future Work

We are careful to note several issues with our experimental setup and exposition. We use ResNet50 as an example of a deep CNN. However, this was trained with 1000 classes that include many other non-bird classes. Specifically, ResNet50 includes chainlink_fence, windows_screen, and jigsaw_puzzle. The seam carving moves the bird images closer in appearance to these classes. If the deep CNN was restricted to only bird images, it may not be so easily fooled. Future work is to train a more modest CNN on a restricted set of classes and evaluating other ImageNet categories. Future work also includes using different energy functions such as histogram of gradients and entropy (Avidan and Shamir, 2007).

5 CONCLUSIONS

In this paper, we presented the seam doppelganger approach for privacy preserving images. The paper detailed how redundant pixels can be identified using the seam carving technique and then replaced. This produced distorted, though still human recognisable, images. We also demonstrated how the image can be restored to a close facsimile of the original, though with some objectionable artefacts. We then showed how the distorted images degraded the accuracy performance of a leading edge image classifier.

ACKNOWLEDGEMENTS

This work was supported in part by the University of Montevallo Contract #19-0501-001. The authors greatly appreciate the support of the staff involved in the project. Without their efforts this research could not have been conducted.

REFERENCES

(Fall 2011 Release). *ImageNet: Explore WordNet Structure*.

Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10–es.

Das, N., Shanbhogue, M., Chen, S.-T., Hohman, F., Chen, L., Kounavis, M. E., and Chau, D. H. (2017). Keeping the bad guys out: Protecting and Vaccinating Deep Learning with JPEG Compression.

Kumar, A., Verma, S., and Mangla, H. (2018). A survey of deep learning techniques in speech recognition. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 179–185.

Liu, Z., Liu, Q., Liu, T., Xu, N., Lin, X., Wang, Y., and Wen, W. (2019). Feature distillation: DNN-oriented JPEG compression against adversarial examples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 860–868.

Moosavi-Dezfooli, S., Fawzi, A., and Frossard, P. (2016). DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582.

Pope, J. (2020 (accessed September 16, 2020)). *Seam Doppelganger on Open Science Foundation*. https://osf.io/gmn97/.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). Imagenet large scale visual recognition challenge.

Sanchez-Matilla, R., Li, C. Y., Shahin Shamsabad, A., Mazzon, R., and Cavallaro, A. (2020). Exploiting vulnerabilities of deep neural networks for privacy protection. *IEEE Transactions on Multimedia*, pages 1–1.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Traore, B. B., Kamsu-Foguem, B., and Tangara, F. (2018). Deep convolution neural network for image recognition. *Ecological Informatics*, 48:257 – 268.