How to Pretrain for Steganalysis

Jan Butora, Yassine Yousfi, and Jessica Fridrich
Binghamton University
Department of Electrical and Computer Engineering
Binghamton, NY 13850
{jbutora1,yyousfi1,fridrich}@binghamton.edu

ABSTRACT

In this paper, we investigate the effect of pretraining CNNs on ImageNet on their performance when refined for steganalysis of digital images. In many cases, it seems that just 'seeing' a large number of images helps with the convergence of the network during the refinement no matter what the pretraining task is. To achieve the best performance, the pretraining task should be related to steganalysis, even if it is done on a completely mismatched cover and stego datasets. Furthermore, the pretraining does not need to be carried out for very long and can be done with limited computational resources. An additional advantage of the pretraining is that it is done on color images and can later be applied for steganalysis of color and grayscale images while still having on-par or better performance than detectors trained specifically for a given source. The refining process is also much faster than training the network from scratch. The most surprising part of the paper is that networks pretrained on JPEG images are a good starting point for spatial domain steganalysis as well.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Image manipulation**; **Neural networks**;

KEYWORDS

Steganalysis, transfer learning, imagenet, JPEG, convolutional neural network

ACM Reference Format:
Jan Butora, Yassine Yousfi, and Jessica Fridrich. 2021. How to Pretrain for Steganalysis. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '21), June 22–25, 2021, Virtual Event, Belgium.*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3437880.3460395

1 INTRODUCTION

Steganography is the art of hiding information in innocuously looking media called covers, such as digital images. Steganalysis on the other hand aims to detect evidence that steganography took place. The most popular choice of steganalyst's detector nowadays is a convolutional neural network (CNN). Recent advances [6, 7, 29] in steganalysis showed that pretraining neural networks on seemingly irrelevant tasks, such as object recognition, and only refining on steganalysis in a downstream dataset is surprisingly beneficial for a range of neural network architectures. This idea of training a network on one task and refining its parameters on a downstream (target) task is typically referred to as transfer learning [4], and was utilized in many fields with great success [20, 31]. Recently, [14] shows the effect of pretraining on artificially generated images rather than natural images. In [19], the authors study transfer learning for steganalysis by pretraining on LSB matching. This inspired us to study whether the pretraining task matters at all or if one could simply pretrain network detectors on arbitrary tasks, provided a large number of images are presented to the network during training.

Pretraining can be thought of as a different way of initializing the network weights. Large computer vision models as well as models used for steganalysis have millions of parameters. Pretraining the model on some classification problem involving natural images or at least images interpretable by humans endows the model with filters capable of extracting the basic elements forming natural images, such as edges, textures, periodic patterns, etc. Naturally, such filters are relevant for steganalysis because the stego signal is essentially a noise modulated by content. Especially in the JPEG domain, the stego signal contains more low-frequency patterns. Thus, when refining the model on a steganalysis task, the model parameters already occupy a rather small subset of the high-dimensional parameter space (the subset of the parameters that are relevant for analyzing natural images), which makes their refinement for a different classification task much faster as long as the task involves again natural images.

Proper initialization of parameters can play a major role in training, especially on difficult tasks, such as steganalysis of small payloads. With randomly initialized parameters, networks trained from scratch typically struggle with convergence. This is why most CNNs for steganalysis are trained with the so-called pair constraint (PC) [3, 26, 27], which forces a cover image and its stego version into the same minibatch. While this indeed helps with convergence, it actually cripples the network's performance in the end. This is because providing pairs of images from different classes that are almost identical might negatively affect Batch Normalization layers. In [29], the authors refined the SRNet[3] without the PC, which significantly boosted its performance. Furthermore, models pretrained on an object recognition task on the ImageNet were successfully refined for steganalysis [6, 7, 29] without the PC. Lastly, in many situations it is beneficial to pretrain CNNs on images embedded with larger payloads before training on images with a small payload [19, 22, 28].

IH&MMSec '21, June 22–25, 2021, Virtual Event, Belgium.
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8295-3/21/06…$15.00
https://doi.org/10.1145/3437880.3460395

In the next section, we study three types of pretraining tasks, treating them as different model initializations for refining on a downstream task, which in our case is steganalysis. In Section 3, we describe the datasets and detectors used for pretraining and the downstream tasks. Section 4 covers the results of our experiments. Conclusion is given in Section 5.

2 PRETRAINING TASKS

In this section, we describe three very different classification tasks for pretraining on ImageNet. All three tasks were trained on color images, which means that the networks trained this way have three-channel filters in the first layer. We can use this later to train on grayscale images as well by simply copying the luminance representation of the image into all three *RGB* channels. With this strategy, only the very last fully connected (FC) layer will have to be initialized with random weights. Note that we would not be able to do it the other way around – to first pretrain on grayscale images and later refine on color images as this would require reinitialization of the filters in the very first layer of the network, which could negatively impact the training.

IN. The goal of the **I**mage**N**et object recognition task is to predict one of the 1,000 possible object classes in the ImageNet database [9].

JIN. **J**-UNIWARD **I**mage**N**et steganalysis task uses ImageNet images as cover images. To generate stego images, we embed only in the luminance channel of the covers with J-UNIWARD [13] with a random payload uniformly distributed in the range $[0.4, 0.6]$ bits per non-zero AC DCT coefficient of the luminance channel (bpnzac). We did not embed into color because our goal was to later use this pretraining on grayscale images as well. This prevents the network to specialize on detecting distortion in the chrominance channels. We believe that exploiting color dependencies later on the downstream task should still be enough. The range of payloads was chosen purely heuristically as 0.4 bpnzac is typically used as a good starting point to train networks from scratch with training on smaller payloads carried out via curriculum training [3, 22]. This again can be thought of as a specific initialization of weights for steganalysis of small payloads. Furthermore, the larger payloads will make the models easier to converge, while additionally diversifying the payload at the same time. We believe the diversification of payload could help with detection on the downstream dataset with different payloads and/or steganographic algorithms.

QIN. **Q**uantization table **I**mage**N**et aims to predict the luminance quantization table as a 64-dimensional vector. While this is a silly task because the quantization table can be read from the JPEG header, the main point of training a model for this task is just parameter initialization when training on some non-obvious task.

3 EXPERIMENTAL SETUP

In this section, we first describe the datasets and the training hyperparameters used for the three pretraining tasks. Then, the datasets and the hyperparameters for the downstream steganalysis tasks are detailed.

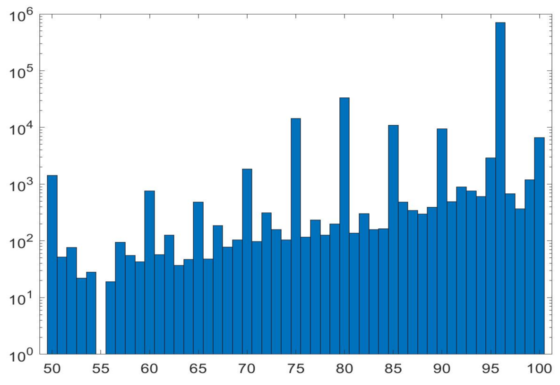


Figure 1: Quality factor distribution in the ImageNet dataset. The histogram is in log-scale.

3.1 Pretraining datasets

Each of the three pretraining tasks uses images from the ImageNet database. There are, however, some task-specific modifications of the images. The original ImageNet consists of over one million color images of varying sizes. The images are JPEG compressed with a variety of quality factors whose distribution is shown in Figure 1. The vast majority of the images are compressed with quality factor 96 (note the logarithmic scale). By inspecting DCT coefficient histograms, we also found strong evidence of multiple compression in some of these images. Therefore, we believe that the majority of these images were actually compressed two or more times with the last compression quality of 96.

For the IN object recognition task, we used training and validation sets of $1,221,405$ and $40,000$ images, respectively. The images during training were augmented with torchvision's *RandomResizedCrop*[1] function with default parameters and size 224×224 and random horizontal flip. The validation images were first rescaled into squares and then resized to 224×224.

The JIN and QIN pretraining tasks use the same dataset generated by cropping ImageNet images to 256×256 pixels. This required us to remove a portion of the images because some of them were already smaller than the target size. We also removed grayscale images. The images were adaptively cropped to avoid singular content (details are provided below). After cropping, we end up with $858,357$ images in the training set and $38,000$ images in the validation set. We did not use a test set, since we only report validation performance on the pretraining task. All cropped images were embedded with J-UNIWARD with random payload as described in Section 2. Both the JIN and QIN tasks used cover and J-UNIWARD stego images. While these datasets are based on the ImageNet, the images we generated are not specifically related to the ImageNet and the same approach can be applied to any dataset.

3.1.1 Smart cropping. For a given luminance channel represented with a DCT matrix of size (M, N), for the (i, j)-th 8×8 block, $i, j \in \{1, \ldots, M/8\} \times \{1, \ldots, N/8\}$, we define the block energy B_{ij} as the number of DCT coefficients $C^{(ij)}_{kl}$, $k, l \in \{1, \ldots, 8\}$ satisfying $|C^{(ij)}_{kl}| > T$: $B_{ij} = \sum_{k,l=0}^{7} [|C^{(ij)}_{kl}| > T]$. We then selected a 256×256

[1] *RandomResizedCrop*

	pretraining		
Model	IN	JIN	QIN
SRNet	0.8524	0.7571	0.7571
B0	0.9325[4]	0.7071	N\A

Table 1: The best validation accuracy on different pretraining tasks. Top5 accuracy is used for IN. Average accuracy of correct prediction is used for JIN and QIN. JIN for B0 is IN+JIN.

Model	Pretraining	J-UNI (0.4)		SI-UNI (0.6)	
		75	95	75	95
SRNet	N\A	$\underline{0.0754}$	$\underline{0.1895}$	$\underline{0.1974}$	$\underline{0.2636}$
	JIN	0.0573	0.1704	0.1751	0.2326
	IN	0.0817	0.2150	$\underline{0.2047}$	$\underline{0.2681}$
	QIN	0.0906	0.2103	0.2319	0.3454
B0	IN+JIN	0.0870	0.2518	0.2242	0.3005
	IN	0.1122	$\underline{0.5000}$	$\underline{0.2462}$	$\underline{0.3624}$

Table 2: P_E of J-UNIWARD at 0.4 bpnzac and SI-UNIWARD at 0.6 bpnzac. Underlined values indicate that the training would not start converging without the PC.

crop from the image that maximizes the energy $\sum_{i,j=0}^{31} B_{n/8+i,m/8+j}$, where m, n are the starting indices of the crop. This was implemented using a convolution between the matrix[2] $[|C_{kl}^{(ij)}| > T]$ and a sliding window of size 256×256 filled with ones. Because a convolution of such scale would be computationally demanding, it was done in the Fourier domain using Matlab's implementation of 2-D Fast Fourier Transform *fft2* and its inverse version *ifft2*, where we computed elementwise multiplication instead of convolution (the sliding window was appropriately padded). To avoid disrupting the JPEG 8×8 block structure, we enforce the cropping to contain only whole 8×8 blocks, meaning that the top left corner corresponds to a DC mode of some block. If there were several crops of the image with the same maximal energy, we simply selected the first such block (in a row-by-row order). After visually inspecting several images for different values of T, $T = 5$ was selected as the smallest value that avoids singular or purely noise content.[3]

3.2 Pretraining schedules

The detector we used in most of our experiments is the SRNet [3]. We used the EfficientNet-B0 [18] only for a subset of experiments with JIN to briefly verify the generalization abilities of the pretraining strategies. In fact, we were not able to make B0 converge on JIN with randomly initialized weights, so we trained it on JIN with weights already pretrained on IN. For every pretraining task, we only need to modify the FC layer of our model because the steganalysis models work with a different number of classes. During every pretraining task, the images fed to the network were loaded into the RGB representation with OpenCV library and rounded to integers to avoid issues with manual chrominance channels upsampling. All trainings were carried out with automatic mixed precision and D_4 augmentations.

[2]The square brackets denote the Iverson bracket.
[3]The code used to generate cropped images is available here
[4]The validation split used to obtain this value contains images that the model was trained on because we did not have access to the data split used for training the CNN.

3.2.1 JIN. Because in JIN the model predict only two cover/stego classes, the FC layer has only two output neurons. The loss function was binary cross entropy loss as it is the standard loss for binary classification problems. We used AdaMax optimizer with eps value 10^{-4} and weight decay 2×10^{-4}. The learning schedule was OneCycle schedule with maximum learning rate (LR) 10^{-3} at epoch 3, division factor 25, and final division factor 10. We tested batch sizes (BS) 64 and 128 with the former performing slightly better on the pretraining task. Hence, we used BS 64 for evaluation of this method. Because the training without the pair constraint would not converge from scratch, we used the PC (32 cover-stego pairs) for the first 10 epochs to allow the network to converge. We continued training without the PC for 50 more epochs, totaling 60 epochs of training. The best checkpoint in terms of the validation accuracy on the pretraining task was achieved after 57 epochs.

B0 was trained with AdamW optimizer with weight decay 10^{-3} with other hyperparameters kept unchanged. The best checkpoint was achieved after 58 epochs.

3.2.2 QIN. Pretraining on QIN was done with the same hyperparameters as on JIN. The only exception is that we never used the PC because it does not make sense in this setting. Thus, the training was carried out for 60 epochs without the PC. We did, however, train it on the same dataset as JIN (including the stego images). The reason being that the steganalyst can always generate stego images and include them in the pretraining task in the same way she uses cover images. Because there are 64 DCT modes and, consequently, values in the quantization table, the FC layer outputs 64 values. For the loss function, we used the average MSE over all quantization steps, minimizing the distance between the (rounded) predicted and the true quantization steps.

3.2.3 IN. Pretraining on object recognition requires quite a few modifications as opposed to the other two tasks under investigation. Using all images from the ImageNet datasets, the FC layer has to have 1,000 output neurons, and we employed the multi-class cross-entropy loss function. Instead of AdaMax, we used the SGD optimizer with eps value 10^{-7} and weight decay 10^{-5}. OneCycle scheduler was used here too but with maximum LR 0.5 at epoch 5 with the other parameters of the scheduler kept the same. Because a larger BS is usually used for object recognition tasks on ImageNet, we used BS 256. Finally, because the training set is roughly twice smaller than in the other two tasks (there are no stego images), we increased the number of epochs to 100.

Instead of pretraining the B0 model on IN, we simply loaded the trained weights available in the pytorch package.[5]

3.3 Downstream datasets

To verify that the pretraining is successful for steganalysis of both color and grayscale images, we verify the proposed pretraining tasks with steganalysis in two downstream datasets.

The first is a typical database used for benchmarking steganography – a union of BOSSbase 1.01 [1] and BOWS2 [2] datasets, both of which contain 10,000 grayscale images resized to 256×256 using Matlab's *imresize*. The detectors were trained on all BOWS2 images and randomly selected 4,000 BOSSbase images. A total of

[5]https://github.com/lukemelas/EfficientNet-PyTorch

1, 000 BOSSbase were used for validation and the remaining 5, 000 for testing. With this database, we benchmark on JPEG and spatial domain steganography. For the JPEG domain, we include in our experiments two quality factors 75 and 95. We used J-UNIWARD at 0.4 and 0.2 bpnzac and its side-informed version SI-UNIWARD [13] at 0.6 bpnzac. In the spatial domain, we used MiPOD [24] and HILL [15] with 0.4 bits per pixel (bpp).

The second dataset is the ALASKA2 dataset [7] used in the recent Kaggle competition.[6] It consists of color images compressed with three quality factors 75, 90, and 95. For every quality, there are 25, 000 images of size 512×512. As in the Kaggle competition, we used stego images generated using UERD [11], J-UNIWARD, and J-MiPOD [8] with the payload spreading strategy across images and color channels as explained in [7].

3.4 Downstream schedules

While refining on the pretraining task, regardless of the downstream dataset, the SRNet was trained with the same hyperparameters that were used during JIN pretraining. Pretrained SRNets were always trained without the PC for 100 epochs unless stated otherwise, while the SRNet with random initial weights was trained with the PC for 300 epochs first and then refined for 100 more epochs without the PC. In some cases, when even the pretrained network would not converge, we added 100 epochs with the PC first. This shows that not all pretraining strategies are equally good starting points for the refinement.

To speed up the training of B0, we used a BS 96 with all other hyperparameters kept as in JIN pretraining. This network was also trained with 100 epochs without the PC. Similarly to SRNet, in some cases the training would not start, in which case the same solution with the PC as above was applied.

4 RESULTS

First, we are going to compare the pretraining strategies on JPEG domain steganography with J-UNIWARD and the side-informed version SI-UNIWARD. To confirm that the results are not specific to J-UNIWARD, which was also used during JIN pretraining, we additionally report the results when detecting steganography in the spatial domain. It seems a bit surprising to use pretraining on JPEG images for spatial-domain steganalysis but, as shown in this paper, the pretraining does work well in this case, too.

For completeness, we include the best validation performance on the pretraining tasks in Table 1. For IN, we measure the performance with Top5 accuracy. For JIN, we used accuracy, and for QIN we also report the average accuracy of correctly predicting the quantization steps when rounding the network output to integers.

4.1 JPEG domain

During training in the JPEG domain, the images were decompressed without rounding to integers. To validate that the pretraining is not skewed toward the dominant quality factor 96 (see Figure 1), we test on two quality factors 75 and 95. Table 2 shows that to detect J-UNIWARD at 0.4 bpnzac, every pretraining used allowed us to train the networks without the PC while achieving a decent performance. One exception was observed for IN-B0 at QF 95, which we were

[6]https://www.kaggle.com/c/alaska2-image-steganalysis

Pretraining	J-UNI (0.2)	
	75	95
N\A (0.4)	0.2076	0.3433
JIN	0.1754	0.3294
IN	<u>0.2059</u>	<u>0.3679</u>
QIN	<u>0.2260</u>	<u>0.3701</u>

Table 3: P_E of SRNet on J-UNIWARD at 0.2 bpnzac. Underlined values indicate that the training would not start converging without the PC. The network trained from scratch was refined from a larger payload of 0.4 bpnzac via payload curriculum training.

not able to train even with the PC. The networks pretrained on IN would not converge with SI-UNIWARD at 0.6 bpnzac, while those pretrained on QIN and JIN would. Even though QIN does not require the PC to train, we see a clear improvement of JIN over QIN pretraining. The overall best performance is achieved with JIN pretraining.

To further validate that JIN pretraining does not allow us to train only on payload 0.4 bpnzac and larger, we include the performance on J-UNIWARD at 0.2 bpnzac in Table 3 and on ALASKA 2 competition setting in Table 4. On 0.2 bpnzac J-UNIWARD, we can see a clear benefit of JIN pretraining over IN or QIN, as it is the only one not requiring the PC to converge. Moreover, it provides the best detection out of the three pretraining tasks. The network trained from scratch on 0.4 bpnzac also converges without the PC, however, JIN performs better even in this case.

In ALASKA 2 setting, the quantity used for measuring the performance is the weighted area under the curve (wAUC) [7]. We do not see much of a difference in performance between IN and JIN for both SRNet and B0. We think this might be due to fact that the training set is sufficiently large and the network architecture cannot get better anymore, a behavior that was already observed in [23]. Nevertheless, one interesting observation remains. Both models were trained on all quality factors without the PC without any need to tweak the training schedule. While this is not surprising for B0 [29], it drastically simplifies the training of the SRNet across multiple quality factors [30]. As reported in [29], when training the SRNet from scratch, it had to be trained first on QF 75 with the PC to allow convergence, then refined without the PC, and finally trained via curriculum learning for the other two qualities.

The last experiment we carried out in both the JPEG and spatial domains investigates how much pretraining is actually needed. The top part of Figure 2 shows the detection error P_E of the SRNet on a downstream task as a function of the number of epochs trained on a pretraining task. Note that for the three downstream tasks: J-UNIWARD at QFs 75 and 95 with 0.4 bpnzac and MiPOD 0.4 bpp, it is enough to pretrain in JIN setting for about 10 epochs. While this is surprising, recall that during those 10 epochs the network is exposed to $10 \times 2 \times 858, 357$ (roughly 17 million) images. Additional pretraining does not provide any significant boost. IN pretraining seems to require more than 20 epochs of pretraining for J-UNIWARD at QF 95. We want to mention at this point that the JIN pretrained model converges even with one pretrained epoch, most likely because the pretraining is done with the PC. In the bottom part of Figure 2, we show the detection error P_E with respect to

Model	Pretraining	UERD 75	UERD 90	UERD 95	J-UNIWARD 75	J-UNIWARD 90	J-UNIWARD 95	J-MiPOD 75	J-MiPOD 90	J-MiPOD 95	Mixture
SRNet	JIN	0.9336	0.9521	0.9444	0.8675	0.8853	0.8894	0.9750	0.9761	0.8572	0.9219
	IN	0.9349	0.9505	0.9369	0.8710	0.8828	0.8851	0.9776	0.9761	0.8595	0.9213
B0	IN+JIN	0.9517	0.9620	0.9500	0.8862	0.8893	0.8949	0.9783	0.9777	0.8625	0.9298
	IN	0.9479	0.9607	0.9486	0.8817	0.8793	0.8836	0.9827	0.9747	0.8546	0.9258

Table 4: wAUC on ALASKA 2 setting.

Model	Pretraining	MiPOD	HILL
SRNet	N\A	<u>0.1554</u>	<u>0.1508</u>
	JIN	0.1396	0.1295
	IN	0.1839	0.1676
	QIN	0.1752	0.1651
B0	IN+JIN	0.2255	0.1971
	IN	<u>0.2736</u>	<u>0.2357</u>

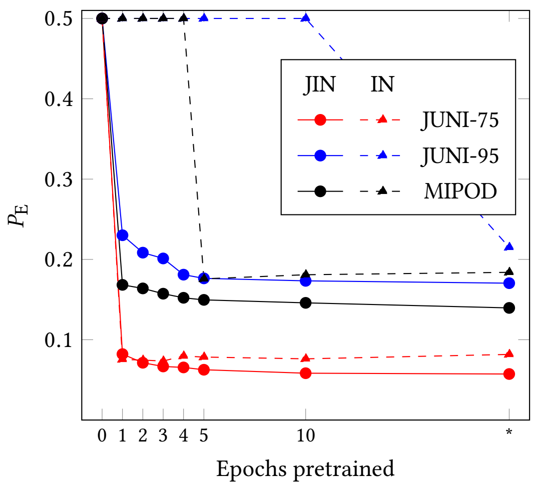
Table 5: Detection error P_E on HILL and MiPOD with 0.4 bpp. Underlined values indicate that the training would not start converging without the PC.

relative accuracy on the pretraining task. We can asses that it is enough to achieve 80−90% of attainable accuracy on the pretraining task in order to have a good initialization of weights for steganalysis.

4.2 Spatial domain

The biggest surprise of this work is not only the fact that the pretraining on JPEG images allows refining the network to detect spatial domain steganography but the fact that we can actually achieve a better detection on uncompressed images embedded with MiPOD and HILL with 0.4 bpp, as seen in Table 5. Additionally, all the pretraining strategies allowed the SRNet to train without the PC, which is puzzling as some of these models were struggling with J-UNIWARD at QF 95 – we expected JPEG steganography to be somewhat easier to train on with pretraining done on JPEG images mostly compressed with QF 96. There is also a significant boost in performance for the JIN pretrained B0 model.

Since we experimented with a more complex dataset in the JPEG domain (ALASKA 2), we decided to test the pretraining effect on a diversified stego source in the spatial domain too – the same source previously used in [5]. The BOSSbase1.01+BOWS2 dataset was embedded with seven algorithms: HILL, WOW [12], S-UNIWARD, MiPOD, non-adaptive and optimally coded LSB matching, Edge-Adaptive (EA) [17], and HUGO [21], all with 0.4 bpp. We then trained the models with the PC because in [5] the performance was measured as the missed-detection error for a fixed false alarm rate (the false alarm 6.44% was used in [5], which we kept in this work too). It was experimentally verified that while training with the PC does not affect a robust measure, such as wAUC, the error at a fixed low false alarm can be dramatically different. Table 6 shows the missed-detection error for the seven algorithms included in the training as well as the wAUC of the detectors. To verify the generalization ability to unseen stego algorithms, we further tested the detectors on pentary MVG [25], MG [10], CMD-HILL [16] with payload 0.4 bpp, and HILL and MiPOD with 0.3 bpp, none of which was included in the training. Once again, a very clear benefit of JIN pretraining is observed for every algorithm whether or not it was used during training.



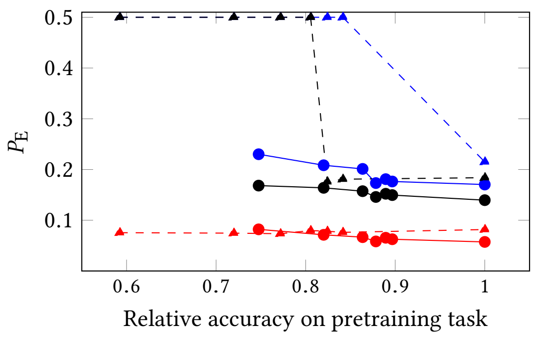


Figure 2: SRNet's detection error P_E at 0.4 bpnzac on BOSSbase1.01+BOWS2. Top: P_E vs. the number of pretrained epochs; IN was pretrained for twice as many epochs because its training set is roughly twice smaller. The superscript * corresponds to the best checkpoint on the pretraining task, which is 57 for JIN and 98 for IN. Bottom: P_E as a function of the relative accuracy w.r.t. the best checkpoint on the pretraining task.

5 CONCLUSIONS

Pretraining of neural networks can be thought of as a non-random initialization of network's parameters. We study different methods of pretraining on a large ImageNet dataset as an initialization for steganalysis. The pretrained models were successfully refined for steganalysis of JPEG images and also for uncompressed images. By studying pretraining on object recognition (1000 classes), on estimation of the quantization table, and on J-UNIWARD, we discovered that all three tasks help with convergence when refining for steganalysis. Pretraining on the steganalysis task, however, provided the best overall results. Moreover, steganalysis-based pretraining

	Pretraining		
	N/A	JIN	IN
HILL	0.2678	**0.2442**	0.2868
WOW	0.1526	**0.1306**	0.1636
S-UNI	0.1646	**0.1382**	0.1812
MiPOD	0.2736	**0.2500**	0.3034
LSBM	**0.0468**	0.0510	0.0494
EA	0.1028	**0.0816**	0.1066
HUGO	0.2376	**0.2324**	0.2560
wAUC	0.9700	**0.9761**	0.9689
MVG	0.2924	**0.2666**	0.3208
MG	0.1696	**0.1414**	0.1774
CMD-HILL	0.4614	**0.4244**	0.4904
HILL (0.3)	0.3796	**0.3538**	0.4074
MiPOD (0.3)	0.4004	**0.3764**	0.4356

Table 6: Missed detection error rate at false alarm 0.0644 **of payload** 0.4 **bpp. Algorithms in bold were not included in the training. Every model trained with the PC.**

always achieves a similar or better performance as detectors trained from scratch on the downstream steganalysis dataset while simplifying the training procedure in diversified stego and cover sources for both grayscale and color images, side-informed schemes, and for small payloads. Pretraining on a steganographic task has an additional benefit of using only two (or several for multiple stego algorithms) classes as opposed to 1000 classes for the ImageNet object recognition task. This allows the pretraining to be carried out with a fairly small mini-batch size, avoiding the need for distributing the training over several GPUs. The pair constraint, which is specific to steganalysis, seems to be unnecessary with a proper initialization (pretraining) of the network parameters.

Future directions include studying the effect of the size of the pretraining/downstream dataset, unifying the detection of steganography in JPEG and spatial domain, pretraining on a large dataset of never compressed images, and investigating the effect of refinement on different parts of the neural network.

ACKNOWLEDGMENTS

The work on this paper was supported by NSF grant No. 1561446.

REFERENCES

[1] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing BOSS. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Conference*, volume 6958 of Lecture Notes in Computer Science, pages 59–70, Prague, Czech Republic, May 18–20, 2011.

[2] P. Bas and T. Furon. BOWS-2. http://bows2.ec-lille.fr, July 2007.

[3] M. Boroumand, M. Chen, and J. Fridrich. Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193, May 2019.

[4] S. Bozinovski and A. Fulgosi. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica 3-121-5*, 1976.

[5] J. Butora and J. Fridrich. Detection of diversified stego sources using CNNs. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2019*, San Francisco, CA, January 26–30, 2019.

[6] K. Chubachi. An ensemble model using CNNs on different domains for ALASKA2 image steganalysis. In *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[7] R. Cogranne, Q. Giboulot, and P. Bas. ALASKA–2: Challenging academic research on steganalysis with realistic images. In *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[8] R. Cogranne, Q. Giboulot, and P. Bas. Steganography by minimizing statistical detectability: The cases of JPEG and color images. In C. Riess and F. Schirrmacher, editors, *The 8th ACM Workshop on Information Hiding and Multimedia Security*, Denver, CO, 2020. ACM Press.

[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255, June 20–25, 2009.

[10] J. Fridrich and J. Kodovský. Multivariate Gaussian model for designing additive distortion for steganography. In *Proc. IEEE ICASSP*, Vancouver, BC, May 26–31, 2013.

[11] L. Guo, J. Ni, W. Su, C. Tang, and Y. Shi. Using statistical image model for JPEG steganography: Uniform embedding revisited. *IEEE Transactions on Information Forensics and Security*, 10(12):2669–2680, 2015.

[12] V. Holub and J. Fridrich. Designing steganographic distortion using directional filters. In *Fourth IEEE International Workshop on Information Forensics and Security*, Tenerife, Spain, December 2–5, 2012.

[13] V. Holub, J. Fridrich, and T. Denemark. Universal distortion design for steganography in an arbitrary domain. *EURASIP Journal on Information Security, Special Issue on Revised Selected Papers of the 1st ACM IH and MMS Workshop*, 2014:1, 2014.

[14] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh. Pre-training without natural images. In *Asian Conference on Computer Vision (ACCV)*, 2020.

[15] B. Li, M. Wang, and J. Huang. A new cost function for spatial image steganography. In *Proceedings IEEE, International Conference on Image Processing, ICIP*, Paris, France, October 27–30, 2014.

[16] B. Li, M. Wang, X. Li, S. Tan, and J. Huang. A strategy of clustering modification directions in spatial image steganography. *IEEE Transactions on Information Forensics and Security*, 10(9):1905–1917, September 2015.

[17] W. Luo, F. Huang, and J. Huang. Edge adaptive image steganography based on LSB matching revisited. *IEEE Transactions on Information Forensics and Security*, 5(2):201–214, June 2010.

[18] T. Mingxing and V. L. Quoc. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, pages 6105–6114, June 9–15, 2019.

[19] S. Ozcan and A. F. Mustacoglu. Transfer learning effects on image steganalysis with pre-trained deep residual neural network model. In *IEEE International Conference on Big Data (Big Data)*, pages 2280–2287, December 10–13, 2018.

[20] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, October 2010.

[21] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In R. Böhme and R. Safavi-Naini, editors, *Information Hiding, 12th International Conference*, volume 6387 of Lecture Notes in Computer Science, pages 161–177, Calgary, Canada, June 28–30, 2010. Springer-Verlag, New York.

[22] Y. Qian, J. Dong, W. Wang, and T. Tan. Learning and transferring representations for image steganalysis using convolutional neural network. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2752–2756, 2016.

[23] H. Ruiz, M. Chaumont, M. Yedroudj, A. O. Amara, F. Comby, and G. Subsol. Analysis of the scalability of a deep-learning network for steganography "into the wild". 2020.

[24] V. Sedighi, R. Cogranne, and J. Fridrich. Content-adaptive steganography by minimizing statistical detectability. *IEEE Transactions on Information Forensics and Security*, 11(2):221–234, 2016.

[25] V. Sedighi, J. Fridrich, and R. Cogranne. Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model. In A. Alattar and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, volume 9409, San Francisco, CA, February 8–12, 2015.

[26] G. Xu. Deep convolutional neural network to detect J-UNIWARD. In M. Stamm, M. Kirchner, and S. Voloshynovskiy, editors, *The 5th ACM Workshop on Information Hiding and Multimedia Security*, Philadelphia, PA, June 20–22, 2017.

[27] M. Yedroudj, F. Comby, and M. Chaumont. Yedroudj-net: An efficient CNN for spatial steganalysis. In *IEEE ICASSP*, pages 2092–2096, Alberta, Canada, April 15–20, 2018.

[28] Y. Yousfi, J. Butora, Q. Giboulot, and J. Fridrich. Breaking ALASKA: Color separation for steganalysis in JPEG domain. In R. Cogranne and L. Verdoliva, editors, *The 7th ACM Workshop on Information Hiding and Multimedia Security*, Paris, France, July 3–5, 2019. ACM Press.

[29] Y. Yousfi, J. Butora, E. Khvedchenya, and J. Fridrich. ImageNet pre-trained CNNs for JPEG steganalysis. In *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[30] Y. Yousfi and J. Fridrich. JPEG steganalysis detectors scalable with respect to compression quality. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2020*, San Francisco, CA, 2020.

[31] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.