Transformers for Pooled Steganalysis

Edgar Kaziakhmedov and Jessica Fridrich, Department of ECE, SUNY Binghamton, NY, USA, {ekaziak1, fridrich}@binghamton.edu

Abstract

In batch steganography, the sender communicates a secret message by hiding it in a bag of cover objects. The adversary performs the so-called pooled steganalysis in that she inspects the entire bag to detect the presence of secrets. This is typically realized by using a detector trained to detect secrets within a single object, applying it to all objects in the bag, and feeding the detector outputs to a pooling function to obtain the final detection statistic. This paper deals with the problem of building the pooler while keeping in mind that the Warden will need to be able to detect steganography in variable size bags carrying variable payload. We propose a flexible machine learning solution to this challenge in the form of a Transformer Encoder Pooler, which is easily trained to be agnostic to the bag size and payload and offers a better detection accuracy than previously proposed poolers.

Introduction

Batch steganography and pooled steganalysis has been introduced by Ker in 2006 [13]. It generalizes the concept of sending a secret embedded in a single object to multiple objects. The sender divides her message into chunks that get embedded in individual cover objects in the bag and then all sent to the recipient. This is called batch steganography. On the other hand, the Warden is allowed to inspect the entire bag to decide whether steganography is being used (pooled steganalysis).

Since the inception of this concept, researchers studied techniques for spreading the payload (assigning the payload chunks to individual objects) as well as Warden strategies for detecting the covert communication [17, 19, 16, 14, 15, 12, 10, 8, 21, 24, 25, 27, 23, 28]. While the communicated objects can be text, digital media files, and other objects, most research went into the scenario when secrets are hidden in digital images. Trivial payload spreading strategies include the uniform sender that spreads the payload equally among all images in the bag, the greedy sender that fully embeds a subset of images, and senders that spread the chunks based on the image content. This last payload spreading strategy includes the image merging sender (IMS) [24], which uses a content-adaptive steganographic algorithm to spread the payload, and the shift-limited and minimum deflection senders [27] that spread the payload based on the feedback from a detector trained to detect steganography.

Pooled steganalysis is typically implemented by pooling the soft outputs of a single image detector (SID) applied to all images in the bag. For detecting modern content-adaptive steganography, SIDs are typically trained on examples of cover and stego images embedded with a range of payloads. The pooling function can have many forms but can be broadly divided into two groups—analytic and data driven poolers. The simplest analytic poolers include the simple average pooler, which computes the arithmetic average of all SID outputs, and the max pooler that computes order statistic of the outputs. Pooling functions can also be derived from first principles based on a statistical model of the SID output on cover and stego images [8, 27, 13]. Finally, poolers can also be built with machine learning tools with bag size independent representations of SID's outputs [23, 28]. In this paper, we show how transformers can be used for building pooling functions that are agnostic to the bag size as well as the payload while providing better performance than previous art.

This paper is structured as follows. In the next section, we briefly discuss relevant prior art and then formally introduce the basic concepts used in batch steganography and pooled steganalysis—the spreading strategy and the pooler. Next, we introduce machine learning poolers and the Transformer Encoder Pooler (TEP). Finally, we compare the detection performance of the TEP with previous art and summarize the paper.

Prior art

The first work on data-driven pooling functions is due Pevný et al. [23]. The authors studied the simplest batch steganography strategies, the uniform and greedy senders, and a uniform sender on a fraction of the images from the bag. The learned pooling function was a linear classifier operating on Parzen window representations of SID outputs. For a fixed known payload and one bag size of 100, the learned pooler was experimentally shown to be very close to the simple average pooler. The authors hypothesized that better pooling functions might be obtained by employing non-linear classifiers and by using higher dimensional representations of the images rather than scalar (SID) outputs. A similar approach to pooled steganalysis has been adopted in the experimental work by Zakaria et al. [28].

Recently, the authors of [9] used Random Forest (RF) for pooling the outputs of a SID for the joint task of detecting batch steganography and source biasing. To this end, the pooler accepted the SID soft outputs as well as the slopes of the so-called detector response curves. The pooler was trained separately for each bag size and communication rate in an effort to use the best performing pooler.

While methods that represent bags with bag size independent representations, such as histograms of SID outputs, enjoy low implementation complexity, they have limitations already pointed out in [23]. First, the pooler might

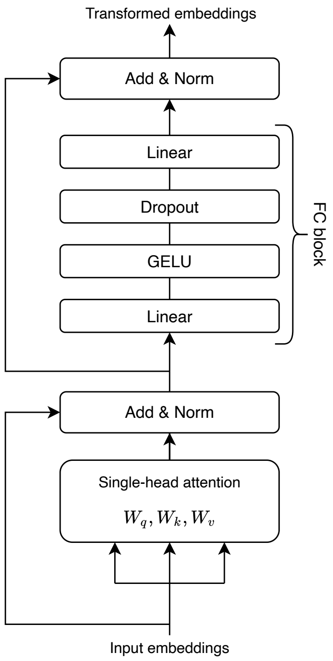

Figure 1. *Transformer encoder block. Positional embeddings are omitted. For simplicity, multi-head attention is replaced with a single-head. The symbols W_q, W_k, and W_v stand for query, key, and value matrices used in the self-attention block.*

be improved if allowed to work with more information extracted from each image than the SID output. Second, due to the data processing inequality [6] the histogram (Parzen window) representation will necessarily degrade the detection accuracy, which may especially become pronounced for small bags, which are likely to be sent in practice.

In contrast, the proposed TEP offers several advantages: TEP can be trained to be bag size and communication rate agnostic and it gracefully scales to higher-dimensional bag representations. Moreover, TEP can accept supplementary information, such as estimated payloads embedded in each image and other image metadata, including the JPEG quality factor, ISO settings, etc. This flexibility in incorporating diverse and multimodal information sets the transformer approach apart from previous art.

Batch steganography and pooled steganalysis

In this section, we introduce the key technical concepts we will need to explain our approach to pooled steganalysis. We start by formally introducing the concept of batch steganography and then describe analytic as well as machine learning pooling functions that will be used in this paper.

Let us denote the set of all possible grayscale cover images X with a fixed number of N pixels as $\mathcal{X}$. Furthermore, let $\mathbf{X} = (X_1, \ldots, X_n)$ be a bag of n cover images. Assuming the sender uses a ternary steganographic scheme, up to $\log_2 3$ bits per pixel (bpp) can be embedded in each image.

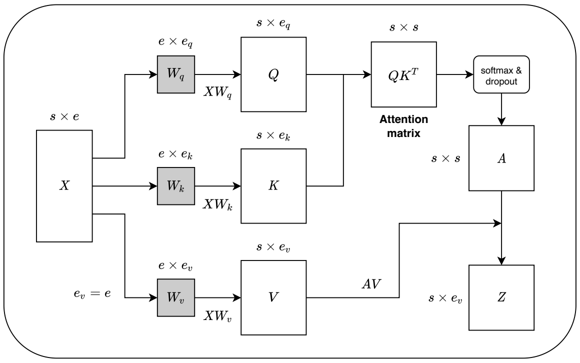

Figure 2. *Single-head attention block. X is a matrix of s embedding vectors of length e. Matrices W_q, W_k, and W_v are set to be trainable. The output matrix Z is a matrix of n transformed embeddings of length e_v. Typically, $e_v = e$.*

To embed a secret payload of rnN bits in bag $\mathbf{X}$, where $0 \leq r \leq \log_2 3$ bpp is the rate, a spreading strategy is used to assign relative payloads $0 \leq \alpha_i \leq \log_2 3$ to each image X_i, $i = 1, \ldots, n$. The α_i must satisfy $r = \frac{1}{n}\sum_{i=1}^{n} \alpha_i$.

A single image detector (SID) is a mapping $d : \mathcal{X} \to \mathbb{R}$. For d implemented as a linear classifier, $d(X)$ is the projection of some inner representation of X on the classifier weight vector. For d in the form of a CNN, $d(X) \in [0, 1]$ is the soft output (logit) after `softmax`. Having intercepted a bag of n images $\mathbf{Y} = (Y_1, \ldots, Y_n)$, a pooler that operates on SID outputs is a mapping $\pi : \mathbb{R}^n \to \mathbb{R}$

$$\pi\left(d(Y_1), \ldots, d(Y_n)\right). \tag{1}$$

As already pointed above, a pooler could conceivably utilize more information from each image than just the scalar output of the SID. A higher-dimensional feature could be extracted from each image and the pooler could operate on a concatenation of such features extracted from all images in the bag. The section "Machine learning poolers" below discusses several choices for the feature vector.

Analytic poolers

The simplest analytic pooler is the average of SID outputs

$$\pi_{\mathrm{AVG}}(d(\mathbf{Y})) = \frac{1}{n}\sum_{i=1}^{n} d(Y_i), \tag{2}$$

where we used for brevity $d(\mathbf{Y}) \triangleq (d(Y_1), \ldots, d(Y_n))$. Note that π_{AVG} is agnostic of the spreading strategy and the communication rate r.

For a Warden aware of the spreading strategy and communication rate r, we also consider the correlator pooler [27, 9]

$$\pi_{\mathrm{CORR}}(d(\mathbf{Y})) = \frac{1}{n}\sum_{i=1}^{n} d(Y_i)\hat{\alpha}_i, \tag{3}$$

where $\hat{\alpha_i}$ is the estimated payload in the ith image, inferred by the Warden from the images at hand. When $\alpha_i = \hat{\alpha}_i$

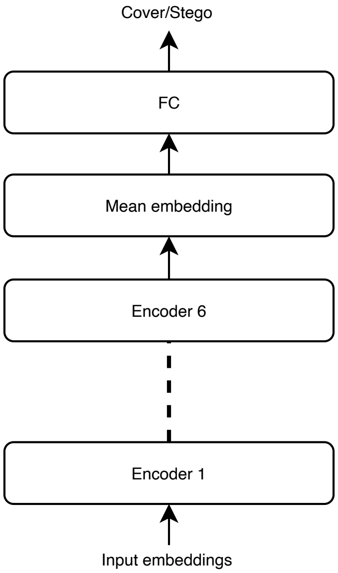

Figure 3. *Transformer Encoder Pooler (TEP) architecture. TEP consists of 6 encoder blocks. Output embeddings are averaged and the resulting embedding vector is passed to a fully-connected layer.*

for all i, the pooler is clairvoyant. In all our experiments, we use this clairvoyant correlator pooler as the worst-case scenario for the sender and also to lower the computational complexity of the experiments. Since in this work we use the IMS spreading strategy, the clairvoyant pooler is a feasible assumption since the embedding costs of most modern content adaptive embedding schemes are largely insensitive to the embedding changes themselves—the payloads estimated from the stego images are nearly identical to the payloads computed from covers [7].

While analytic poolers are by design agnostic to the bag size, they may not perform the best in practice. In contrast, machine learning-based poolers may achieve better detection accuracy by learning more complex patterns from the training data.

Machine learning poolers

Pooling functions can certainly be built with Machine Learning (ML) techniques. Most importantly, they can make use of more complex representations of the bags. In this paper, we use higher dimensional internal representations of images learned within a trained CNN.

Formally, a trained CNN SID d consists of an "internal feature extractor" $f : \mathcal{X} \to \mathbb{R}^m$ followed by an MLP/IP layer $\ell : \mathbb{R}^m \to \mathbb{R}$

$$d(Y) = \ell(f(Y)). \tag{4}$$

For example, the popular CNN architecture SRNet uses a 512-dimensional output of the last convolutional layer before it is processed by the IP layer. This vector could be used for more complex bag representations to train a ML pooler.

In this paper, we will use three different representations of a bag for training ML poolers :

1. The bag is represented with soft SID output only
$$\mathbf{f}_d = d(\mathbf{Y}) \in \mathbb{R}^n. \tag{5}$$
2. The bag is represented with internal features
$$\mathbf{f}_f = (f(Y_1), \ldots, f(Y_n)) \in \mathbb{R}^{m\times n}. \tag{6}$$
3. Payloads can be added to the representation if they are known or estimable. Both types of representations from 1) and 2) can be extended by appending the vector of n payloads $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ embedded in each image
$$\mathbf{f}_{d,\alpha} = (\mathbf{f}_d, \boldsymbol{\alpha}) \in \mathbb{R}^{2n} \tag{7}$$
$$\mathbf{f}_{f,\alpha} = (\mathbf{f}_f, \boldsymbol{\alpha}) \in \mathbb{R}^{(m+1)\times n}. \tag{8}$$

Once the feature vectors are extracted for all bags, a machine learning tool can be used to train a pooler for a fixed bag size n. To cover all possible bag sizes, one could possibly train a pooler for each bag size. While this is obviously cumbersome, expensive, and not suitable for practical applications, such poolers have previously been used for research purposes [27, 9]. In this paper, we train Random Forest [3] (RF) and Logistic Regression [22] (LogReg) classifiers, which were selected for their ability to handle high-dimensional features while maintaining low training complexity.

ML poolers implemented with a bag size independent representation of images were introduced in [23]. To make the feature vector size invariant to the bag size, the authors used the following Parzen Window (PW) mapping

$$\mathbf{f}_h = \left(\frac{1}{n}\sum_{i=1}^{n} k\big(d(Y_i), c_1\big), \ldots, \frac{1}{n}\sum_{i=1}^{n} k\big(d(Y_i), c_p\big)\right) \in \mathbb{R}^p, \tag{9}$$

where $k(x,y) = \exp(-\gamma(x-y)^2)$ is the Gaussian kernel and $\{c_i\}_{i=1}^p$ is a set of equally spaced points in a real interval bounded by the range of $d(Y)$ ($[0,1]$ for CNN outputs). A payload-informed PW pooler works with the feature vector $\mathbf{f}_{h.\alpha}$

$$\mathbf{f}_{h,\alpha} = \left(\frac{1}{n}\sum_{i=1}^{n} k\big(d(Y_i), c_1\big), \ldots, \frac{1}{n}\sum_{i=1}^{n} k\big(d(Y_i), c_p\big),\right. \tag{10}$$
$$\left.\frac{1}{n}\sum_{i=1}^{n} k\big(\alpha_i, c'_1\big), \ldots, \frac{1}{n}\sum_{i=1}^{n} k\big(\alpha_i, c'_p\big)\right) \in \mathbb{R}^{2p},$$

where $\{c'\}_{i=1}^p$ is a set of equally spaced points in the interval $[0, \log_2 3]$.

Transformers

The transformer architecture was first introduced in [26], where the authors proposed a simpler and more

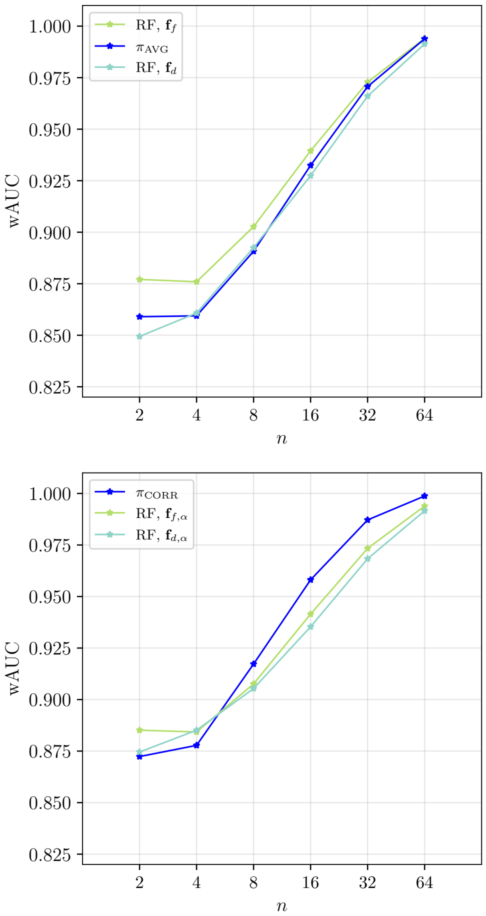

Figure 4. *Detection accuracy wAUC of RFs trained on different representations of bags as a function of the bag size n for a fixed rate $r = 0.4$. The top plot shows the RF trained on internal SID features $\mathbf{f}_f$ vs. RF trained on just soft outputs $\mathbf{f}_d$. The bottom plot contrasts the performance of the RF when adding the payloads ($\mathbf{f}_{d,\alpha}$ and $\mathbf{f}_{f,\alpha}$). Both subplots also contain the corresponding analytic poolers, π_{AVG} and π_{CORR}.*

efficient architecture for sequence modeling to eliminate convolutions and recurrence. The transformer utilizes encoder-decoder structure similar to many other neural sequence transduction models. The baseline transformer architecture proposed in [26] consists of 6 encoder and 6 decoder blocks.

Figure 1 shows a simplified architecture of an encoder, comprising a self-attention block followed by a skip connection (Add) and normalization (Norm), as well as a fully connected block, which is also followed by a skip connection and normalization. The fully connected block consists of a first fully connected layer with a customizable hidden layer size, an activation function, a dropout layer, and a second fully connected layer. For the activation function, we employ the Gaussian Error Linear Unit (GELU) [11], which exhibits increased curvature and non-monotonicity, enabling it to approximate complex functions more effectively than ReLU. Additionally, we incorporate a dropout layer within the fully connected block to mitigate overfitting. Note that this simplified architecture excludes positional encoding, which is crucial for text sequence modeling but irrelevant for pooled steganalysis since images in a bag could have an arbitrary order.

The core element of the encoder is a self-attention block, which allows modeling long-range dependencies across input embeddings. Figure 2 shows how the self-attention block is built. It consists of trainable matrices W_q, W_k, and W_v (query, key, and value matrices). Matrices W_q and W_k transform input embeddings into the attention matrix QK^T which indicates how each embedding relates to the other embeddings. Eventually, the attention matrix is passed through `softmax` and weighted by the V matrix to form the final transformed embeddings. This process generates refined embeddings that capture contextual relationships within the input sequence. Similar to the fully connected block, we incorporate dropout to mitigate overfitting. We note that the original transformer encoder uses multi-head attention block to speed up the computation; we illustrate a single-head attention block for simplicity.

In pooled steganalysis, embeddings are feature vectors obtained for each image in a bag. For example, we extract s embeddings of length e for each image (e depends on the bag representation), then each bag could be represented as a sequence of embeddings forming an $s \times e$ embedding matrix. The transformer architecture allows us to process input embedding to capture relationships across embeddings vectors. Multiple encoder blocks permit capturing more high-level dependencies. However, the transformed embeddings are still bag size dependent. In order to remove the dependency on the bag size, we take an average of transformed embeddings forming a single embedding vector of length e. Eventually, the embedding vector is passed to a fully connected layer for a binary classification task. The proposed architecture is shown in Figure 3. We experimented with different amount of encoders and set it to 6 as it achieves the best performance while being computationally efficient.

The proposed TEP architecture has multiple advantages. TEP is a data-driven pooler that is agnostic to the bag size. In other words, bags of different sizes could be used for training and for inference with a single network. In addition, the embedding vectors could be of any nature. Specifically, they can contain image metadata, which has a potential to improve the detection accuracy. Most importantly, a trained TEP could be applied to unseen bag sizes and communication rates due to the generalization ability of neural networks. All these aspects make the TEP a flexible solution for pooled steganalysis. In the following section, we extensively test the performance of the proposed TEP and contrast it against state-of-the-art poolers.

Experiments

In this section, we describe the experimental setup and evaluate the TEP against alternative pooling methods.

All experiments were conducted using the ALASKA II dataset [5] prepared as described in [5] but without the final JPEG compression. The entire dataset comprises 75,000

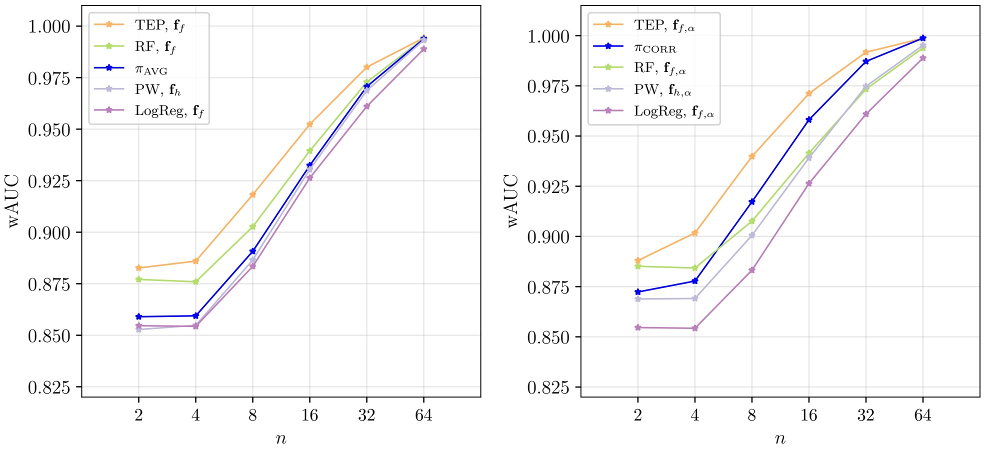

Figure 5. *Detection accuracy wAUC of different poolers across variable bag sizes with a fixed communication rate $r = 0.4$. The poolers are trained with bag representations $\mathbf{f}_f/\mathbf{f}_h$ (left) and $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ (right).*

images randomly partitioned into three disjoint subsets (for compatibility with our previous work): Split A, Split W, and Split T. Split A contains 25,000 images and serves as the training set for Alice's SID. Within Split A, we allocate 22,000 images for training, 1,000 for training, and 2,000 for testing. To construct the bags for evaluation, we utilize Split T which also consists of 25,000 images. Given that some poolers require separate image sets for training and testing, we randomly assign one-third of Split T to training (Split Ta) and and reserve the remaining two-thirds for evaluating the poolers (Split Tb). Split W with 25,000 images is unused.

In our experiments, we use a SID implemented as an SRNet [1] trained on cover and stego images from Split A with weights seeded from JIN-SRNet [4]. This means that the internal feature representation (as explained in Section "Machine learning poolers"), $f(Y) \in \mathbb{R}^m$, $m = 512$, is the output of the last convolutional layers in the trained SRNet (SID) before the IP layer when presenting it with image Y on the input.

The stego images were created with HILL [20] and with relative payloads in bpp sampled uniformly from

$$\mathcal{P} = \{0.05, 0.1, 0.2, ..., 1.4, 1.5\}. \tag{11}$$

Stego bags are generated using the Image Merging Sender (IMS) [24], which is a well-studied approach that naturally extends single-image steganography to the bag setting. The IMS treats a bag of n images, each containing N pixels, as a single large image, into which a total payload of rnN bits is embedded. Notably, the embedding costs are precomputed on individual images using the HILL cost function [20]. The actual embedding is simulated on the rate–distortion bound with an embedding simulator.

For training data-driven poolers, it is essential to generate disjoint sets of bags for training, validation, and testing. To this end, we construct 10,000 pairs of cover/stego bags from Split Ta and 5,000 pairs from Split Tb for evaluation. The first 10,000 pairs are further partitioned into 8,000 pairs for training and 2,000 pairs for validation. All benchmarking is conducted on the testing set using bags of size $n \in \{2, 4, 8, 16, 32, 64\}$ with the performance evaluated using the weighted Area Under the Curve (wAUC) [5].

The training process for machine learning-based poolers follows a standardized pipeline. All input features are normalized to zero mean and unit variance. Prior to training, we conduct hyperparameter estimation for each machine learning algorithm. For Random Forest (RF), the tuned hyperparameters include the number of estimators (300, 500, and 700), the maximum number of features (`None`, `sqrt`, `log2`), the maximum depth (`20`, `32`, `None`), the minimum number of samples to split an internal node (`2`, `5`, `10`), and the minimum number of samples required to be at a leaf node (`1`, `2`, `4`). For Logistic Regression (LogReg), the search space includes the solver parameters (`liblinear`, `saga`), the inverse of regularization strength (from `-2.5` to `10`), and the maximum number of iterations (`1,000`, `5,000`, `10,000`). Hyperparameter tuning is performed via a Monte Carlo cross-validation procedure with three iterations, using a fixed 1:3 train-test split ratio. All training and evaluations were implemented in Python with the `scikit-learn` package.

The TEP is implemented using `PyTorch` library and is publicly available in our GitHub repository[1]. We explore various architectural configurations, including the number of encoder blocks, attention heads, feature dimensionality per head, hidden layer size, and dropout rate. Through empirical evaluation, we determine that the optimal configuration consists of 6 encoder blocks, 8 attention heads, a feature dimensionality of 64 per head, a hidden layer size of 1024, and a dropout rate of 0.1. For training, we implement a custom batch sampler to ensure that all bags within a batch are of the same size. Additionally, to prevent the network from relying on the order of the images, all features within a bag are randomly permuted at each step. The optimization is performed using the `AdamW` optimizer with a `one-cycle` learning rate scheduler with an initial

[1] https://github.com/DDELab/TEP

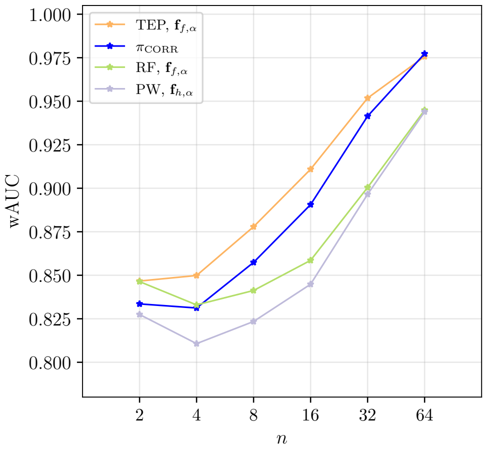

Figure 6. *Detection accuracy wAUC of different poolers across variable bag sizes with a fixed communication rate $r = 0.3$ and $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ feature vector.*

learning rate of 10^{-3}. Notably, in all experiment the TEP is always trained exclusively on bags of sizes $\{2,4,8\}$, meaning that larger bag sizes $\{16,32,64\}$ are never encountered during training. In contrast, the RF and LogReg poolers are trained separately for each bag size.

In the following subsections, we benchmark TEP alongside other poolers in two distinct setups. The first setup involves testing on bags of varying sizes, while keeping the communication rate fixed to assess the effect of the bag representation on detection accuracy. We also assess whether the TEP can effectively detect bags of unseen sizes that are embedded with a known communication rate. In the second setup, both the bag size and the communication rate vary. This allows us to evaluate TEP's ability to maintain robust detection performance across bags with unseen sizes and varying communication rates.

Effect of bag representation

First, in Figure 4 (top) we demonstrate that training a machine learning pooler (RF and LogReg) on internal feature vectors $\mathbf{f}_f \in \mathbb{R}^{512\times n}$ (Eq. (6)) rather than solely on the soft outputs $\mathbf{f}_d = d(\mathbf{Y}) \in \mathbb{R}^n$ leads to improved detection performance. The bottom subplot further illustrates that utilizing internal feature vectors with payloads $\mathbf{f}_{f,\alpha}$ enhances performance compared to $\mathbf{f}_{d,\alpha}$. These experiments were executed for a fixed communication rate of $r = 0.4$ across all bag sizes. We note that a separate RF pooler was trained for each tested bag size.

Comparing ML poolers

Having established that the SID's internal features constitute a superior choice for representing the bag, we proceed to investigate which pooler achieves the optimal performance and whether incorporating payloads further enhances it. To this end, we train RF, LogReg, TEP, and PW poolers using both $\mathbf{f}_f/\mathbf{f}_h$ and $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ feature vectors. We remind the reader that the RF and LogReg classifiers had to be trained for each bag size as the dimensionality of the features $\mathbf{f}_f$ and $\mathbf{f}_{f,\alpha}$ depends on the bag size n.

To determine the hyper parameters γ and p for the PW bag representations (Eq. (9)), we experimented with various kernel width parameters γ and observed that the optimal value computed as described in [2] (Eq 11) yields the best results. The best performing value of the parameter p was determined as $p = 64$. It is worth noting that while the PW feature vector maintains a fixed size, the PW pooler can be trained in two distinct modes. In the first mode, the PW pooler is trained for each bag size (analogous to RF and LogReg), whereas in the second mode, the PW pooler is trained on data encompassing all bag sizes. Our observations indicate that training the PW pooler with a fixed bag size performs consistently better than when training on all bag sizes.

Figure 5 illustrates the detection performance of various poolers when utilizing $\mathbf{f}_f/\mathbf{f}_h$ (left) and $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ (right) bag representations. The results presented in the plot suggest that incorporating payloads generally leads to improved performance across all poolers with the exception of LogReg for which the performance remains unchanged. Furthermore, the TEP consistently outperforms all other poolers, irrespectively of the chosen feature vector. It is worth noting that RF consistently surpasses LogReg in performance; consequently, RF is adopted as the default machine learning pooler in subsequent experiments. Interestingly, while RF demonstrates an improvement over the simple average pooler, it fails to outperform the correlator pooler for larger bag sizes. This limitation is likely due to RF's potential inability to effectively learn a complex, non-linear decision boundary when confronted with higher input dimensionalities. Finally, the performance of PW is observed to be quite similar to that of RF once the bag size becomes sufficiently large. This behavior is anticipated, as the bins in PW tend to be noisier for smaller bag sizes.

In Figure 6, we show the results of a similar experiment as above for a different rate of $r = 0.3$ and only for the bag representations that include the payloads $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$. Consistent with our previous finding, the TEP exhibits the best performance. While the RF outperforms the correlator for the two smallest bag sizes, it falls behind for the remaining bag sizes. Moreover, we observe a similar trend with the PW pooler, which gradually approaches the performance level of the RF for the largest bag sizes.

In summary, our experiments so far have demonstrated that the TEP achieves the best performance across variable bag sizes for two communication rates, notably despite being trained on only half of the considered bag sizes.

Variable bag size and communication rate

In this section, we evaluate ML poolers on bags of varying sizes where the communication rate decays with the bag size n according to the formula

$$r(n) = \left(\frac{2}{n}\right)^{\lambda} \times 0.8. \qquad (12)$$

Here, the parameter λ controls the rate of decay with the initial payload set to $r(2) = 0.8$. We note that $\lambda = 1/2$

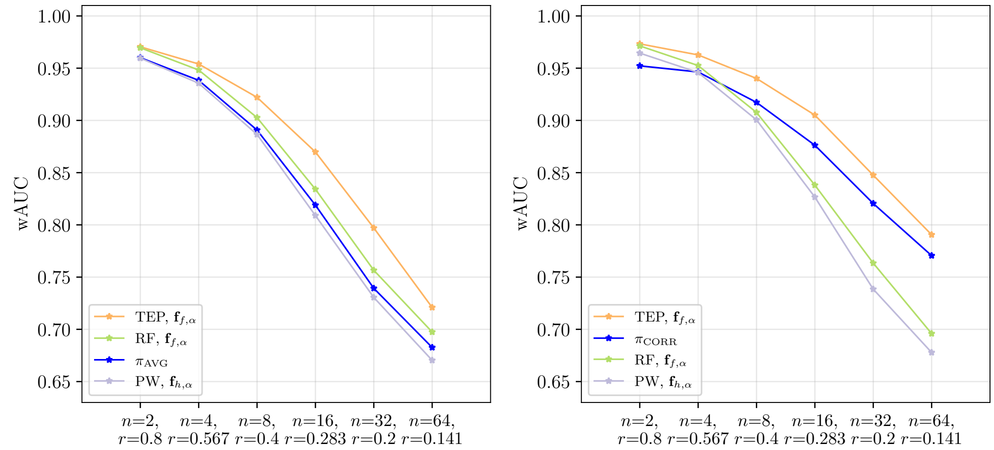

Figure 7. *Detection accuracy wAUC of different poolers across variable bag sizes with decaying communication rate $r(n)$ with $\lambda = 1/2$. Poolers are trained with bag representations $\mathbf{f}_f/\mathbf{f}_h$ (left) and $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ (right).*

corresponds to scaling the payload size according to the square root law [18] (SRL) to guarantee constant statistical detectability across all bag sizes. For $\lambda > 1/2$, the payload size is adjusted with n in a "sub-SRL" fashion meaning that the detectability decreases to zero as $n \to \infty$, while $\lambda < 1/2$ corresponds to super-SRL payload scaling that asymptotically guarantees perfect detection.

In the spirit of the previous benchmarking experiments, we first identify the feature vector that yields the best performance. Figure 7 presents a comparison of the TEP, RF, and PW poolers under the decaying communication rate scenario with $\lambda = 1/2$. The results consistently show a similar trend across all poolers: training with the $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$ bag representations gives better performance compared to other representations. The TEP exhibits the superior detection on both seen/unseen bag sizes and communication rates. Note that the statistical detectability is expected to approach a finite value, as dictated by the SRL. We do not observe this convergence in our experiment because the size of the bags is too small to exhibit the asymptotic behavior.

Next, having selected the feature vector, we vary the parameter λ to induce a slower (super-SRL) or faster (sub-SRL) decay of the communication rate. In Figure 8, we compare the same set of poolers for a decaying rate $\lambda = \frac{1}{4}$ (top) and $\lambda = \frac{3}{4}$ (bottom). In both scenarios, the TEP consistently achieves the best detection accuracy among all tested poolers. We also observe that with $\lambda = \frac{1}{4}$, the detection performance of PW and RF remains relatively stable while the TEP and the correlator attain near perfect detection as one would expect for super-SRL payload scaling.

In summary, our experiments involving variable bag sizes and communication rates have consistently demonstrated that the Transformer Encoder Pooler outperforms all other state-of-the-art pooling functions considered in this study.

Conclusions

This paper addresses the topic of learning pooling functions for steganalysis of digital images. We leverage the transformer, a recently proposed machine learning architecture known for its efficient handling of sequential data, and introduce the Transformer Encoder Pooler (TEP). The TEP has several important advantages over previously proposed machine learning pooling functions. First, it is a flexible design that allows training on bags of varying sizes and varying communication rates. Second, the training complexity gracefully scales with increased dimensionality of bag representation for the pooling function. The ability of the TEP to accept higher-dimensional bag representations gives it a significantly better detection accuracy when compared to pooling just the scalar detector outputs. Third, the architecture is flexible enough to permit representations augmented with image metadata, such as estimated payloads residing in each image, JPEG quality, and ISO setting to potentially further improve detection. The TEP enjoys low training complexity and can detect batch steganography in bags of unseen sizes embedded with a range of communication rates.

Looking ahead, we aim to explore whether TEP can be adapted for batch steganography in the JPEG domain and investigate whether incorporating image metadata to bag representations further enhances detection performance.

Acknowledgments

The work on this paper was supported by NSF grant No. 2324991.

References

[1] 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.

[2] M. Boroumand and J. Fridrich. Boosting steganalysis with explicit feature maps. In F. Perez-Gonzales,

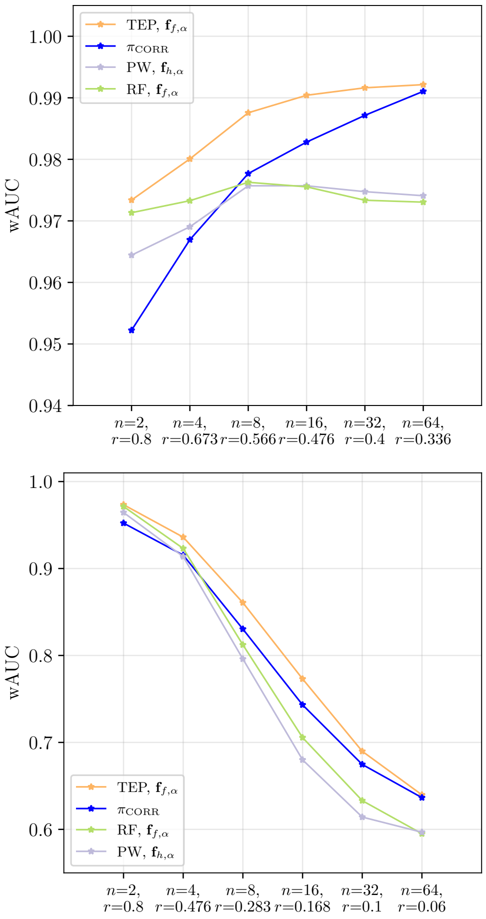

Figure 8. *Detection accuracy wAUC of different poolers across variable bag sizes with decaying communication rate $r(n)$ with $\lambda = 1/4$ (top) and $\lambda = 3/4$ (bottom). The poolers are trained with bag representations $\mathbf{f}_{f,\alpha}/\mathbf{f}_{h,\alpha}$.*

F. Cayre, and P. Bas, editors, *4th ACM IH&MMSec. Workshop*, Vigo, Spain, June 20–22, 2016.

[3] L. Breiman. Random forests. *Machine Learning*, 45:5–32, October 2001.

[4] J. Butora, Y. Yousfi, and J. Fridrich. How to pretrain for steganalysis. In D. Borghys and P. Bas, editors, *The 9th ACM Workshop on Information Hiding and Multimedia Security*, Brussels, Belgium, June 22–25, 2021. ACM Press.

[5] 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.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: John Wiley & Sons, Inc., 1991.

[7] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich. Selection-channel-aware rich model for steganalysis of digital images. In *IEEE International Workshop on Information Forensics and Security*, Atlanta, GA, December 3–5, 2014.

[8] E. Dworetzky and J. Fridrich. Explaining the bag gain in batch steganography. *IEEE Transactions on Information Forensics and Security*, 18:3031–3043, 2023.

[9] E. Dworetzky, E. Kaziakhmedov, and J. Fridrich. Improving steganographic security with source biasing. Baiona, Spain, 2024. ACM Press.

[10] E. Dworetzky, B. Tondi, M. Barni, and J. Fridrich. Payload allocation in batch steganography: A game-theoretic perspective. *IEEE Signal Processing Letters*, 2025. Under review.

[11] D. Hendrycks and K. Gimpel. Gaussian error linear units (GELUs), 2023. Available at `https://arxiv.org/abs/1606.08415`.

[12] X. Hu, J. Ni, W. Zhang, and J. Huang. Efficient JPEG batch steganography using intrinsic energy of image contents. *IEEE Transactions on Information Forensics and Security*, 16:4544–4558, 2021.

[13] A. D. Ker. Batch steganography and pooled steganalysis. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of Lecture Notes in Computer Science, pages 265–281, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[14] A. D. Ker. Batch steganography and the threshold game. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 04 1–13, San Jose, CA, January 29–February 1, 2007.

[15] A. D. Ker. A capacity result for batch steganography. *IEEE Signal Processing Letters*, 14(8):525–528, 2007.

[16] A. D. Ker. Perturbation hiding and the batch steganography problem. In K. Solanki, K. Sullivan, and U. Madhow, editors, *Information Hiding, 10th International Workshop*, volume 5284 of Lecture Notes in Computer Science, pages 45–59, Santa Barbara, CA, June 19–21, 2008. Springer-Verlag, New York.

[17] A. D. Ker. Locally square distortion and batch steganographic capacity. *International Journal of Digital Crime and Forensics*, 1(1):29–44, 2009.

[18] A. D. Ker. The square root law of steganography. 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. ACM Press.

[19] A. D. Ker and Tomas Pevný. Batch steganography in the real world. In J. Dittmann, S. Craver, and S. Katzenbeisser, editors, *Proceedings of the 14th ACM Multimedia & Security Workshop*, pages 1–10, Coventry, UK, September 6–7, 2012.

[20] 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.

[21] L. Li, W. Zhang, C. Qin, K. Chen, W. Zhou, and N. Yu. Adversarial batch image steganography against CNN-based pooled steganalysis. *Signal Processing*, 181:107920–107920, 2021.

[22] P. McCullagh and J. A. Nelder. *Generalized Linear*

Models. Chapman & Hall / CRC, London, 1989.

[23] T. Pevný and I. Nikolaev. Optimizing pooling function for pooled steganalysis. In *IEEE International Workshop on Information Forensics and Security*, pages 1–6, Rome, Italy, November 16–19, 2015.

[24] V. Sedighi, R. Cogranne, and J. Fridrich. Practical strategies for content-adaptive batch steganography and pooled steganalysis. In *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, March 5–9, 2017.

[25] M. Sharifzadeh, M. Aloraini, and D. Schonfeld. Adaptive batch size image merging steganography and quantized Gaussian image steganography. *IEEE Transactions on Information Forensics and Security*, 15:867–879, 2020.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[27] Y. Yousfi, E. Dworetzky, and J. Fridrich. Detector-informed batch steganography and pooled steganalysis. In J. Butora, C. Veilhauer, and B. Tondi, editors, *The 10th ACM Workshop on Information Hiding and Multimedia Security*, Santa Barbara, CA, 2022. ACM Press.

[28] A. Zakaria, M. Chaumont, and G. Subsol. Pooled steganalysis in JPEG: how to deal with the spreading strategy? In *IEEE International Workshop on Information Forensics and Security*, pages 1–6, Delft, Netherlands, December 9–12, 2019.

Author Biography

Edgar Kaziakhmedov received M.S. degree in Applied Mathematics and Physics from Moscow Institute of Physics and Technology, Moscow, in 2020. He is currently pursuing PhD degree in electrical engineering at Binghamton University. His research areas lie within digital image steganalysis and steganography, neural network based image processing and digital media forensics.

Jessica Fridrich is Distinguished Professor of Electrical and Computer Engineering at Binghamton University. She received her PhD in Systems Science from Binghamton University in 1995 and MS in Applied Mathematics from Czech Technical University in Prague in 1987. Her main interests are in steganography, steganalysis, and digital image forensics. Since 1995, she has received 23 research grants totaling over $13 mil that lead to more than 230 papers and 7 US patents.