

Hiding Video in Audio via Reversible Generative Models

Hyukryul Yang*
HKUST

Hao Ouyang*
HKUST

Vladlen Koltun
Intel Labs

Qifeng Chen
HKUST

Abstract

We present a method for hiding video content inside audio files while preserving the perceptual fidelity of the cover audio. This is a form of cross-modal steganography and is particularly challenging due to the high bitrate of video. Our scheme uses recent advances in flow-based generative models, which enable mapping audio to latent codes such that nearby codes correspond to perceptually similar signals. We show that compressed video data can be concealed in the latent codes of audio sequences while preserving the fidelity of both the hidden video and the cover audio. We can embed 128x128 video inside same-duration audio, or higher-resolution video inside longer audio sequences. Quantitative experiments show that our approach outperforms relevant baselines in steganographic capacity and fidelity.

1. Introduction

Consider an activist who needs to publicize a video record of human rights violations under a repressive regime. The regime monitors communication in and out of the country. How can the activist transmit the video without detection? The field of steganography investigates techniques for hiding information within media such as images, video, and audio. Steganography aims to enable concealment of secret content inside publicly transmitted files [29].

In this work, we consider the possibility of hiding video content inside audio files. This pushes the boundaries of steganography, which commonly deals with hiding text messages or embedding media of the same type as the cover file (e.g., images within images) [25]. We choose to conceal video due to its effectiveness in depiction and communication (e.g., the 1992 Los Angeles riots were sparked by video footage of police brutality). We choose audio as the cover medium because audio has higher embedding capacity than text or image files, and because audio sharing platforms such as CLYP and YourListen will not transcode the audio files, which eases the embedding of content inside the file [37].

Hiding video in audio while preserving the fidelity of both the secret and the cover media is extremely challenging. Consider concealing a one-second 128×128 color video in one-second audio with $22K$ samples. There are $128 \times 128 \times 30 \times 24 = 12M$ bits in one second of video, an order of magnitude more than the audio samples. Although a variety of traditional and deep learning methods for steganography have been proposed [3, 41], a direct application of these techniques to our setting would require more than five minutes of audio to absorb one second of video. We aim for much more efficient embedding, such as hiding a video clip in an audio segment of the same length. (One second of video inside one second of audio.)

Our steganography scheme for hiding video in audio builds upon flow-based generative models. Specifically, we use WaveGlow, a reversible generative model that computes bijective mappings between audio signals and latent variables [33]. Although this model is invertible, the latent variable reconstructed from encoded audio may not be exactly the same. Some bits in the reconstructed latent variable may be flipped due to numerical errors in floating-point arithmetic. Therefore, we apply a novel optimization-based strategy to convert binary codes to latent variables. The optimization takes into consideration the average flip rate of each bit in a latent variable and the importance of each bit in a binary code.

We conduct experiments to evaluate the performance of our proposed model and several learning-based and heuristic baselines. Our optimized flow-based model significantly outperforms other baselines in video reconstruction quality and capacity. Our model can efficiently hide a 128×128 video in a same-duration audio file with a sampling rate of 22,050 Hz. The concealed video can be recovered at high fidelity (MS-SSIM 0.965) while the modification to the cover audio is unnoticeable to human listeners. Our approach can also embed one second of high-resolution 848×480 video in a 10-second audio file. Our contributions can be summarized as follows:

- We study a new cross-modal steganography task: hiding video in audio. The encoded audio signals are perceptually indistinguishable from the original.
- We propose a new steganography scheme based on

*Joint first authors

deep reversible generative models.

- We design a novel optimization-based strategy that enables hiding longer video in audio without compromising the quality of the reconstructed video.

2. Related work

2.1. Steganography

Steganography methods can be analyzed in terms of transparency, capacity, and robustness [29]. Transparency indicates that the encoded file is perceptually indistinguishable and is undetectable by steganalysis [29]. Capacity is defined as the total amount of hidden data [6]. Robustness measures the ability to preserve the secret information when intended or unintended modification occurs [1].

Researchers have proposed a variety of steganography methods that use different cover files such as images, audio signals, and video frames to embed secret information [25, 11, 35, 30, 19]. A classic audio steganography method is hiding the secret data in the least significant bits (LSB) [4, 1], so that the subtle changes in audio are not auditorily apparent. However, this alters the statistical distribution of the cover media, resulting in reliable detection by steganalysis. LSB has relatively high capacity but is vulnerable to modification and easy to detect [13]. Furthermore, for hiding video in audio, the capacity of LSB (one or two bits per sample) is far from sufficient. Our proposed method achieves high perceptual transparency and high capacity via a different approach.

Some audio steganography methods such as echo hiding [16] and tone insertion [2] exploit limitations of the human auditory system (HAS) by adding echo signal or low power tones which are less salient to human perception. The encoded audio is perceptually transparent but not secure to detection. Other methods such as phase coding [12] and spread spectrum [9] utilize the phase information of the audio signal, which makes the encoded audio more robust to modification and compression while getting lower capacity.

Deep networks have recently been applied to steganography, with a strong focus on images [3, 41, 17, 34, 31]. Hayes et al. [17] proposed to train an end-to-end deep network for hiding one image inside another. Baluja et al. [3] suggested utilizing an adversarial loss [5, 23] to generate better encoded images. Zhu et al. [41] applied an adversarial loss and increased robustness by training with noise layers and differentiable compression layers. Since the human visual system (HVS) is less sensitive than the human auditory system (HAS), hiding secret information in digital audio is generally more difficult than embedding it in images [40]. Audio generated by adversarial networks may introduce additional noise [10]. We therefore exploit the possibility of hiding a large amount of information in audio using recent advances in deep generative modeling.

2.2. Flow-based Generative Models

A flow-based generative model is constructed by a sequence of invertible transformations to building a one-to-one mapping between a simple prior distribution (i.e. Gaussian) and a complex distribution. Recent flow-based models [8, 33, 26, 21, 32, 27] have produced high-quality results for both image and audio generation. Dinh et al. [7, 8] proposed a novel differentiable and invertible affine coupling layer that serves as a basic transformation in a flow-based network. Kingma and Dhariwal [21] further proposed using a 1×1 convolutional layer in the structure to enhance information propagation. WaveGlow [33] adapted flow-based image generative models to audio generation by using dilated convolutions [39] conditioned on Mel-spectrograms. Our proposed steganography scheme for hiding video in audio leverages the invertibility of WaveGlow. We can hide information in the latent variable in the highest-density regions of the prior distribution, such that the generated audio remains perceptually indistinguishable from the original.

Although there are other types of generative models such as generative adversarial networks (GAN) [15] and variational autoencoders (VAE) [22], flow-based models are more suitable to our framework, which requires an encoder that produces the original latent variable. GANs have no encoder and VAEs do not recover the exact original latent variable since the encoder in a VAE outputs a distribution for sampling latent variables.

3. Preliminaries

3.1. Image Compression Networks

An image compression network can encode an RGB image as a binary code that can be then decoded to reconstruct the image. For a video, we can compute the binary code B of each video frame I by $B = E_I(I)$, where E_I is the encoder. Then we can obtain the reconstructed video frame \hat{I} by $\hat{I} = D_I(B)$, where D_I is the decoder.

We use the image compression network (ICN) proposed by Toderici et al. [36] to encode an image I (e.g., a single frame in a video) because this model is robust to bit corruption. Note that some bits in the reconstructed binary code may be flipped due to numerical errors. The ICN progressively encodes residual information r_k in K iterations:

$$\begin{aligned} r_0 &= I, \\ b_k &= E_I(r_{k-1}, g_{k-1}), \\ r_k &= r_{k-1} - D_I(b_k, h_{k-1}), \\ k &= 1, 2, \dots, K \end{aligned}$$

where g_k and h_k are hidden states in the ConvLSTM model [18]. The encoder and decoder have the same architecture for all K iterations. The model is trained by minimizing the sum of all the residual information $\sum_{k=1}^K \|r_k\|_1$. We can

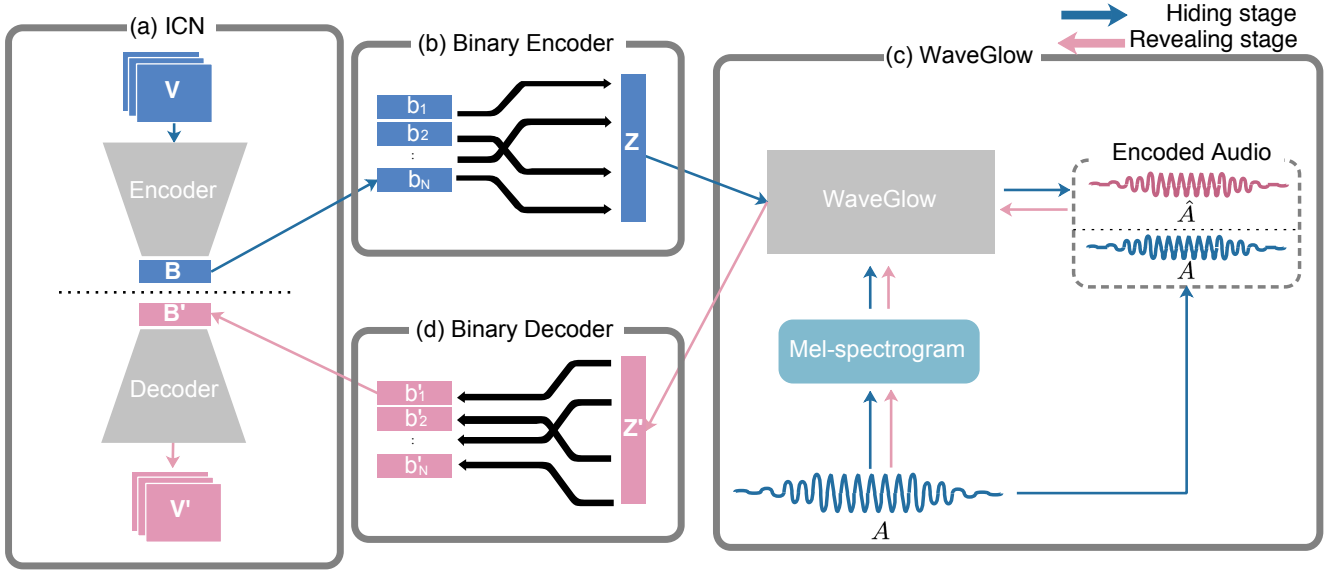


Figure 1: Overview of our steganography pipeline. (a) In the hiding stage, given a video V , the image compression network (ICN) first compresses each frame into a binary code B . (b) The binary encoder embeds a binary code into a latent variable Z . (c) Given the latent variable Z and the mel-spectrogram of cover audio A , WaveGlow generates encoded audio, \hat{A} , which sounds perceptually equivalent to A . In the revealing stage, we first recover a latent variable Z' , leveraging the invertibility of WaveGlow. We then use the binary decoder to reconstruct the binary code B' . Finally, we reconstruct the video V' by feeding B' into the decoder of ICN.

increase the iteration number K to preserve more details, but this creates more binary bits in the binary code. In practice, we set the iteration number K to be 4 to balance the reconstructed image quality and the number of bits.

3.2. WaveGlow

We will use WaveGlow [33] as a key building block in our steganography scheme. WaveGlow is a flow-based generative model that generates audio by sampling from a simple prior distribution (zero-mean spherical Gaussian). The model is composed of a sequence of invertible and differentiable layers in a neural network. The size of output audio signals is the same as the input latent variable. We can formulate the WaveGlow model as follows:

$$\begin{aligned} Z &\sim N(0, I), \\ x &= f_0 \circ f_1 \circ f_2 \dots \circ f_k(Z), \\ Z &= f_k^{-1} \circ f_{k-1}^{-1} \dots \circ f_0^{-1}(x). \end{aligned}$$

The model is trained by minimizing the negative log-likelihood of the data distribution. Since the transformation operations are all bijective, we can apply the change of variables theorem to calculate the log-likelihood directly:

$$\log p_\theta(x) = \log p_\theta(z) + \sum_{i=1}^k \log |\det(J(f_i^{-1}(x)))|.$$

The second term is derived from the change of variables theorem, where J is the Jacobian Matrix and \det is the determinant. This indicates that for each transformation layer, the Jacobian matrix should be easy to compute. With this requirement, two simple but efficient transformations are applied in the network.

Affine coupling layers. Affine coupling layers enable interactions across spatial dimensions. Their architecture is defined as follows:

$$\begin{aligned} x_a, x_b &= \text{split}(x), \\ (\log s, t) &= WN(x_a, \text{mel-spectrogram}), \\ x'_b &= s \odot x_b + t, \\ f_{coupling}^{-1}(x) &= \text{concat}(x_a, x'_b). \end{aligned}$$

The *split* and *concat* operations are defined along the channel dimension. WN uses dilated convolutions with *tanh* and takes mel-spectrograms to control the content and tone of the output audio.

1×1 invertible convolution. 1×1 convolution is used to generate permutation along the channel dimension, which increases the generative capacity of the model.

In our steganography pipeline, we embed the binary code from the compression model into the latent variable Z and use this latent variable to generate encoded audio conditioned on the mel-spectrogram of the input audio. Given the encoded audio, we can retrieve the latent variable by

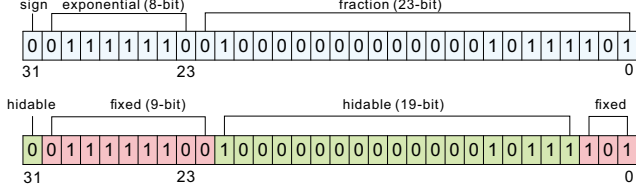


Figure 2: The upper layout describes the structure of the IEEE Standard for Floating-Point Arithmetic (IEEE 754). In the bottom layout, we can hide binary bits in the hidable slots in the IEEE 754 structure.

applying invertible inference in WaveGlow with the same mel-spectrogram.

4. Method

4.1. Overview

Our model consists of a hiding stage and a revealing stage, as illustrated by the blue and pink flowchart arrows in Figure 1. Consider a sequence of video frames V and a cover audio signal A . In the hiding stage, our goal is to hide V into an encoded audio signal \hat{A} that is perceptually indistinguishable to A . In the revealing stage, our goal is to reconstruct \hat{V} from \hat{A} so that \hat{V} is visually similar to V .

In the hiding stage, we hide the binary codes generated by the image compression network (ICN) [36] into an audio signal. Each frame I is compressed into a binary code $B \in \{-1, 1\}^{K \times \frac{H}{16} \times \frac{W}{16} \times 32}$, where K denotes the number of iterations used in ICN. Then the binary encoder embeds all the bits in B into a latent variable Z . Then, a pretrained WaveGlow is used to generate an encoded audio \hat{A} given Z and the mel-spectrogram of signal A . Recall that WaveGlow consists of a sequence of invertible transformations. To reconstruct Z from the encoded audio \hat{A} , we also need the mel-spectrogram of A , and thus we generate a final encoded stereo audio (\hat{A}, A) . Since \hat{A} and A are perceptually indistinguishable, the stereo audio sounds the same as the original audio A .

In the revealing stage, we first reconstruct the latent variable Z' from \hat{A} and the mel-spectrogram from A by WaveGlow. Note that due to numerical errors in floating-point arithmetic, Z' is usually not exactly equal to Z . Then, for each video frame, a binary code B' can be recovered from Z' by the binary decoder. Finally, B' can be used to reconstruct a video frame through the decoder of ICN.

4.2. Binary encoder and decoder

The role of the binary encoder is to embed binary codes into Z (a sequence of 32-bit floating-point numbers), and the binary decoder is an inverse mapping that recovers the embedded binary codes from Z . Since the pretrained

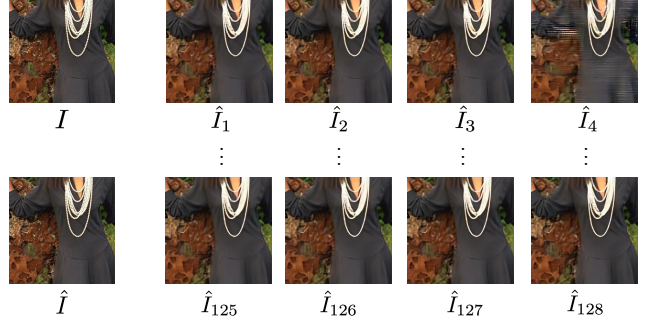


Figure 3: Sample images of \hat{I}_j when the bit b_j is flipped. We can see that flipping \hat{I}_4 is more damaging than other samples. This implies that b_4 is a more important bit in the binary code for image reconstruction.

WaveGlow takes a latent variable sampled from Gaussian normal distribution, the binary encoder should generate floating-point numbers that “follow” the Gaussian distribution. WaveGlow can generate desirable encoded audio as long as Z is in the highest density range of the Gaussian distribution (and the mel-spectrogram of the cover audio).

We utilize the structure of IEEE Standard for Floating-Point Arithmetic (IEEE 754) [14] as shown in Figure 2. A floating-point number β contains 32 positions $\{\beta_i\}_{i=0}^{31}$ where the position index starts from the end of fraction part. The actual value of a floating-point number z can be calculated by

$$\beta = (-1)^{\beta_{31}} \times 2^{(e-127)} \times (1 + m),$$

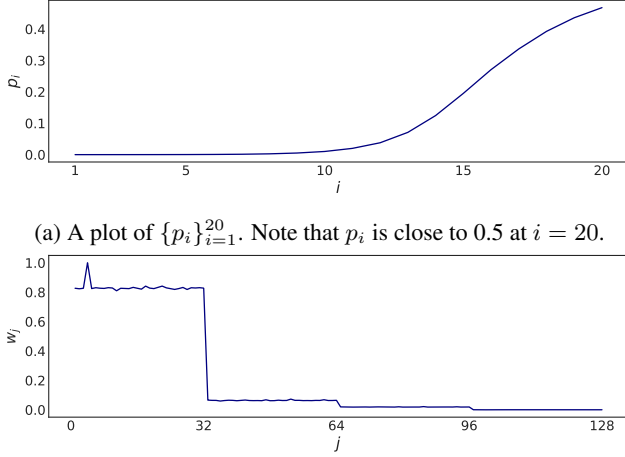
$$e = \sum_{i=0}^7 \beta_{23+i} 2^i,$$

$$m = \sum_{i=1}^{23} \beta_{23-i} 2^{-i}.$$

If we fix 9 positions $\{\beta_i\}_{i=22}^{30}$ to 011111100, the range of β becomes $-0.75 \leq \beta \leq -0.5$ or $0.5 \leq \beta \leq 0.75$. We use 20 positions in β hidable slots, represented as a list $\mathcal{S} = (\beta_3, \beta_4, \dots, \beta_{21}, \beta_{31})$. Note that the three positions $\{\beta_0, \beta_1, \beta_2\}$ are not used because of their high sensitivity to numerical instability.

4.3. Encoding optimization

A simple encoding strategy is to embed every 20 bits in B into Z sequentially, but this is not optimal. Some bits of the recovered binary code B' from Z' can be flipped due to numerical errors in floating-point arithmetic. When we hide 20 bits into a latent variable, the average flip rate is 13.28%. Flipping some bits in the binary code can severely damage the reconstructed video frames. Therefore, we have designed an embedding optimization strategy for better video



(a) A plot of $\{p_i\}_{i=1}^{20}$. Note that p_i is close to 0.5 at $i = 20$.
(b) A plot of $\{w_j\}_{j=1}^{128}$ normalized to $[0, 1]$. We plot the bit importance for 128 bits when $K = 4$. The values drop when j is multiple of 32. There is little difference in bit importance after iteration 4.

Figure 4: Analysis on flip rates and bit importance.

reconstruction quality.

If we can estimate the importance of each bit in B for image reconstruction, the embedding process can be optimized by assigning important bits into slots with low flip rates. The shape of a binary code tensor B for a video frame I_t is $(K, \frac{H}{16}, \frac{W}{16}, 32)$ where K denotes the number of total iterations used in compression, and H and W are the height and width of I_t .

Consider all the bits in $B(\cdot, y, x, \cdot)$ at location (x, y) as a vector $b = \{b_1, b_2, \dots, b_M\}$ where $M = K \times 32$. This M -bit vector b represents a 16×16 patch in the image. Let w_j be the importance of b_j and p_i be the flip rate of the i -th slot in \mathcal{S} . The flip rate p_i indicates how likely the value of the i -th slot in \mathcal{S} will change numerically in the encoding and decoding process of WaveGlow. Let π be an embedding function that maps b_j to the $\pi(b_j)$ -th slot in \mathcal{S} . We can obtain a better embedding function by solving

$$\begin{aligned} & \arg \max_{\pi} \sum_{j=1}^M w_j (1 - p_{\pi(b_j)}), \\ & \text{subject to } \sum_b \sum_{j=1}^M \mathbf{1}[\pi(b_j) = i] \leq L, \forall i \in \{1, 2, \dots, 20\}, \end{aligned} \quad (1)$$

where $\mathbf{1}$ is the indicator function and L is the number of floating-point numbers we can use. The constraint enforces that the total number of assignments to each slot in \mathcal{S} is not greater than L .

We can further alleviate the negative effects of flipped bits by embedding the same bit multiple times to the same slot location in several floating-point numbers and then tak-

ing the majority vote in the binary decoder. We can extend the objective function with multiple assignments:

$$\begin{aligned} & \arg \max_{\pi} \sum_{j=1}^M w_j (1 - p_{\pi(b_j), \tau(b_j)}), \\ & \text{s.t. } \sum_b \sum_{j=1}^M \tau(b_j) \mathbf{1}[\pi(b_j) = i] \leq L, \forall i \in \{1, 2, \dots, 20\}, \end{aligned} \quad (2)$$

where $\tau(\cdot) \in \{1, 3, 5, \dots\}$ is a function indicating the number of the assignments of a bit, and $p_{i,n}$ is the probability of getting a flipped value when we embed a bit n times on the i -th slot:

$$\text{If } n = 1, p_{i,1} = p_i; \quad (3)$$

$$\text{If } n = 3, p_{i,3} = p_i^3 + \binom{3}{2} p_i^2 (1 - p_i); \quad (4)$$

$$\text{If } n = 5, p_{i,5} = p_i^5 + \binom{5}{4} p_i^4 (1 - p_i) + \binom{5}{3} p_i^3 (1 - p_i)^2. \quad (5)$$

Next, we will go through the details of this optimization problem, including the estimation of the flip rate and the importance of each bit in a binary code, and the details of our optimization algorithm.

Bit importance and flip rates. We can measure the importance of each bit b_j by measuring the reconstructed image quality when the value of b_j is flipped. Let \hat{I}_j be the reconstructed image from the binary code B when b_j is flipped for each 16×16 patch. Figure 3 is the sample result of $\{\hat{I}_j\}_{j=1}^{128}$ where the number of iterations is $K = 4$. We evaluate the change in image quality using MS-SSIM [38]. We can use MS-SSIM to compute the bit importance w_j :

$$w_j = \mathbb{E}_I [\text{MS-SSIM}(I, \hat{I}) - \text{MS-SSIM}(I, \hat{I}_j)].$$

In practice, w_j is estimated by sampling images from the COCO dataset [24].

Similarly, the flip rate p_i can be estimated by sampling audio from the LJ Speech dataset [20] and sampling latent variables from a Gaussian distribution for WaveGlow. The values of w_j and p_i are visualized in Figure 4.

Embedding optimization. To simplify the optimization problem, we perform embedding on a frame-by-frame basis. Suppose we are hiding the binary code $B \in \mathbb{R}^{K \times \frac{H}{16} \times \frac{W}{16} \times 32}$ for a frame I . The number of floating-point numbers needed to hide B is $\lceil \frac{K \times 32}{20} \rceil \times \frac{H}{16} \times \frac{W}{16}$. The capacity of each slot i is also $\lceil \frac{K \times 32}{20} \rceil \times \frac{H}{16} \times \frac{W}{16}$, as shown in Figure 5.

We can apply a greedy algorithm to embed the bits in B . We greedily pick an unassigned bit with the highest importance and assign it an available slot with the lowest flip rate. We repeat this process until all the bits are assigned.

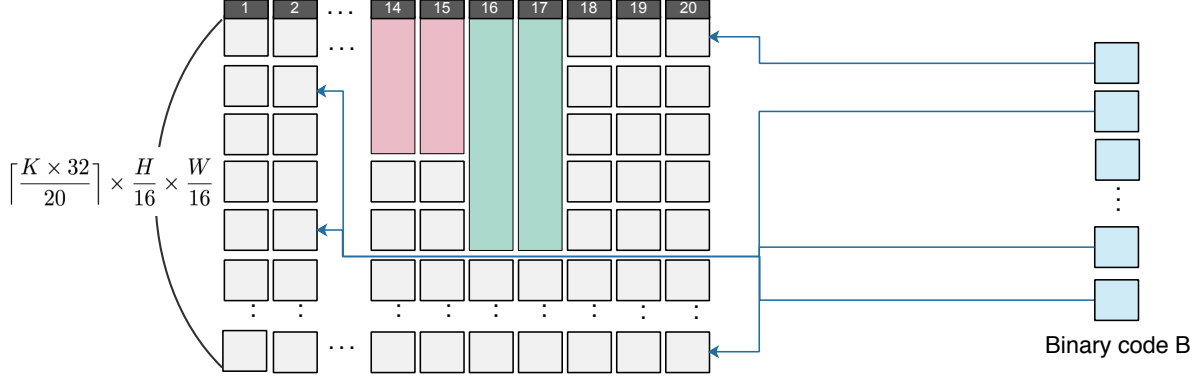


Figure 5: Encoding optimization. We optimize the encoding process by allocating important bits first to slots that have lower flip rates. We assign the same bit multiple times, reconstructing it via majority voting.

Some slots may be unfilled after embedding all the bits in the binary code. We can further exploit this remaining space by assigning the same bit multiple times. We perform a greedy algorithm that picks a bit to be embedded into some slot twice so that the objective in Equation 2 is maximized.

5. Experiments

5.1. Baselines

Naive encoding. We can directly encode a binary code to a latent variable without considering flip rates and bit importance. We can simply hide every 18 bits ($\beta_5, \beta_6, \dots, \beta_{21}, \beta_{30}$) in the binary code into the 18 slots in each floating-point number. Since the size of binary code for each frame is $(K, \frac{H}{16}, \frac{W}{16}, 32)$, we need $\lceil K \times \frac{H}{16} \times \frac{W}{16} \times 32 \times \frac{1}{18} \rceil$ floating-point numbers for embedding one frame.

Continuous bottleneck. We replace the sign operator with a tanh activation in the bottleneck layer of the image compression network [36]. The extracted code from this new network architecture is not binary: it consists of floating-point numbers between -1 and 1. Then we do not need the conversion between binary codes and latent variables. We can directly feed the floating-point code as the latent variable into WaveGlow. The size of the floating-point code is set to be $(K, \frac{H}{16}, \frac{W}{16}, 4)$. The modified image compression network with continuous bottleneck is trained on the COCO dataset [24] for 40 epochs with learning rate $5e - 4$.

ConvNet-based model. Following the deep steganography structure proposed by Baluja et al. [3], we can build a neural network model for hiding a video frame in audio. The model consists of three networks: a preparation network, a hiding network, and a revealing network. We apply the short-time Fourier transform on the cover audio to obtain its spectrogram. Then the preparation network transforms

a secret video frame to an encoded image with the same spatial resolution as the spectrogram. The hiding network takes the spectrogram and the embedded image as input and generates encoded audio that sounds similar to the cover audio. In the end, the revealing network can reconstruct a video from the encoded audio. The model is trained with a video loss that minimizes the difference between the reconstructed video and the original video, and an audio loss that minimizes the difference between the cover audio and encoded audio. The model is trained on the COCO and LJ Speech datasets for 40 epochs with learning rate $5e - 4$.

5.2. Datasets

DAVIS. DAVIS is a popular video segmentation dataset [28]. It consists of 150 videos. Each video contains 69.7 frames on average. For the evaluation of our models and various baselines, we crop all video frames to 128×128 . We also conduct an experiment to embed high-resolution 480p video into audio with our approach.

LJ Speech. We use the LJ Speech dataset [20] for cover audio. This dataset is composed of 13,100 audio clips sampled at 22,050 Hz. Each clip is of length 1 to 10 seconds and reproduces a single speaker reading passages from non-fiction books.

5.3. Hiding 128×128 Video

We first conduct an experiment on hiding all 90 DAVIS videos at resolution 128×128 in audio using our approach as well as all presented baselines. We randomly sample 90 audio clips from the LJ dataset as cover audio. The results are summarized in Table 1. The video reconstructed by our steganography scheme has much high quality than the baselines in terms of PSNR and SSIM. Our method also has the highest capacity in hiding pixels in audio. Our method can hide a 128×128 video in 0.6 seconds of audio.

Methods	PSNR	MS-SSIM	Samples per one-second video
ConvNet (spectrogram)	21.6270	0.9049	330750 (15.00 sec)
Continuous bottleneck	23.2836	0.9384	30720 (1.39 sec)
Naive encoding	11.5683	0.4690	13680 (0.62 sec)
Ours	25.4318	0.9648	13440 (0.60 sec)

Table 1: Quantitative analysis of our model and the baselines. We measure PSNR and MS-SSIM between the ground truth and the reconstructed video frames. The capacity of different models is shown in terms of how many audio samples are needed to hide a one-second video.

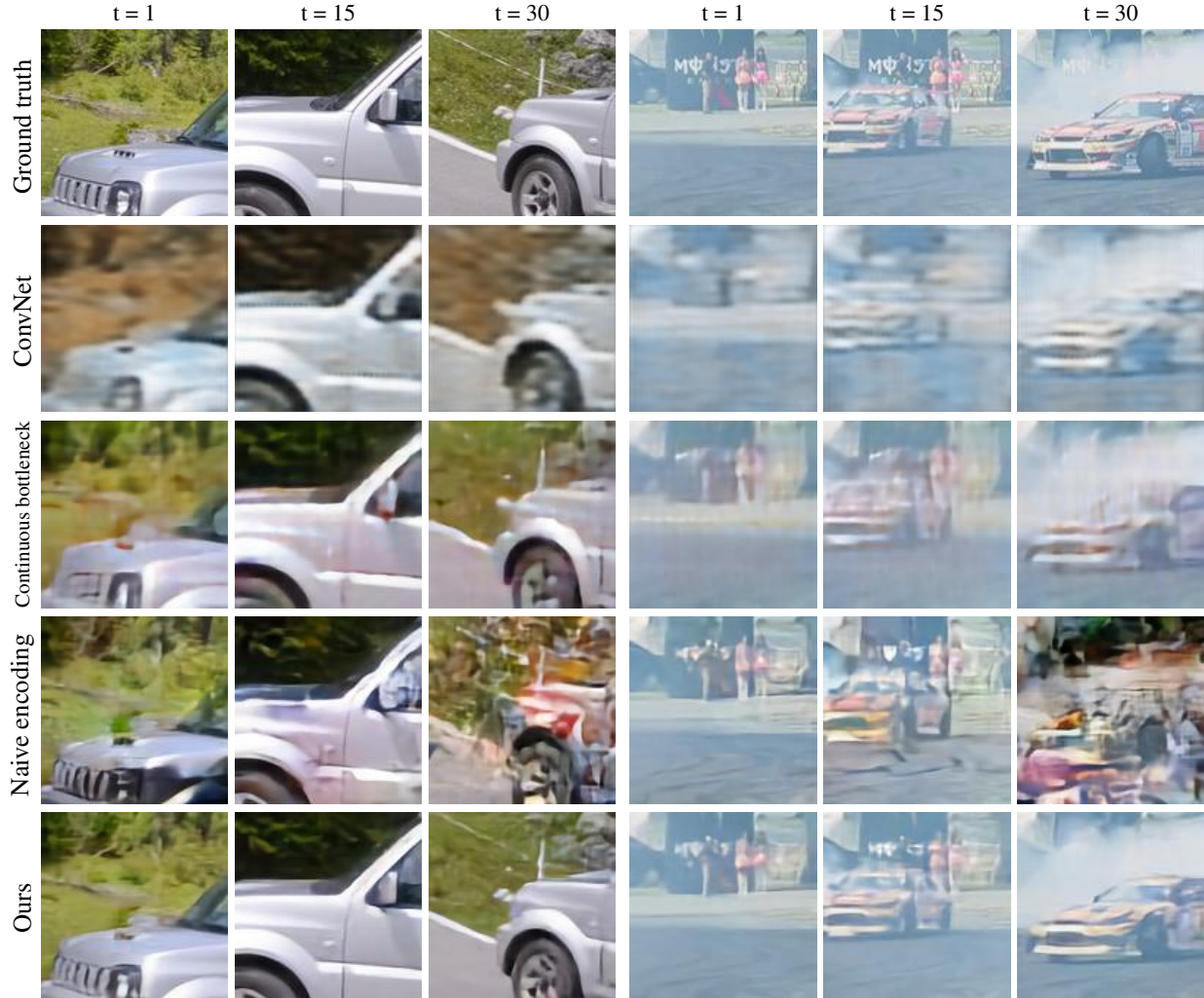


Figure 6: Visual comparison of different methods on hiding DAVIS video [28] in LJ audio [20]. Our approach achieves higher visual quality than the baselines.

Figure 6 provides a qualitative comparison of the visual quality of video frames reconstructed by different methods. Our reconstructed video has higher visual fidelity and fewer artifacts than the baselines.

5.4. Hiding High-resolution Video

Our approach can also hide 480×848 video in audio. Figure 7 visualizes the reconstructed video. Our reconstruction is perceptually indistinguishable from the ground-truth

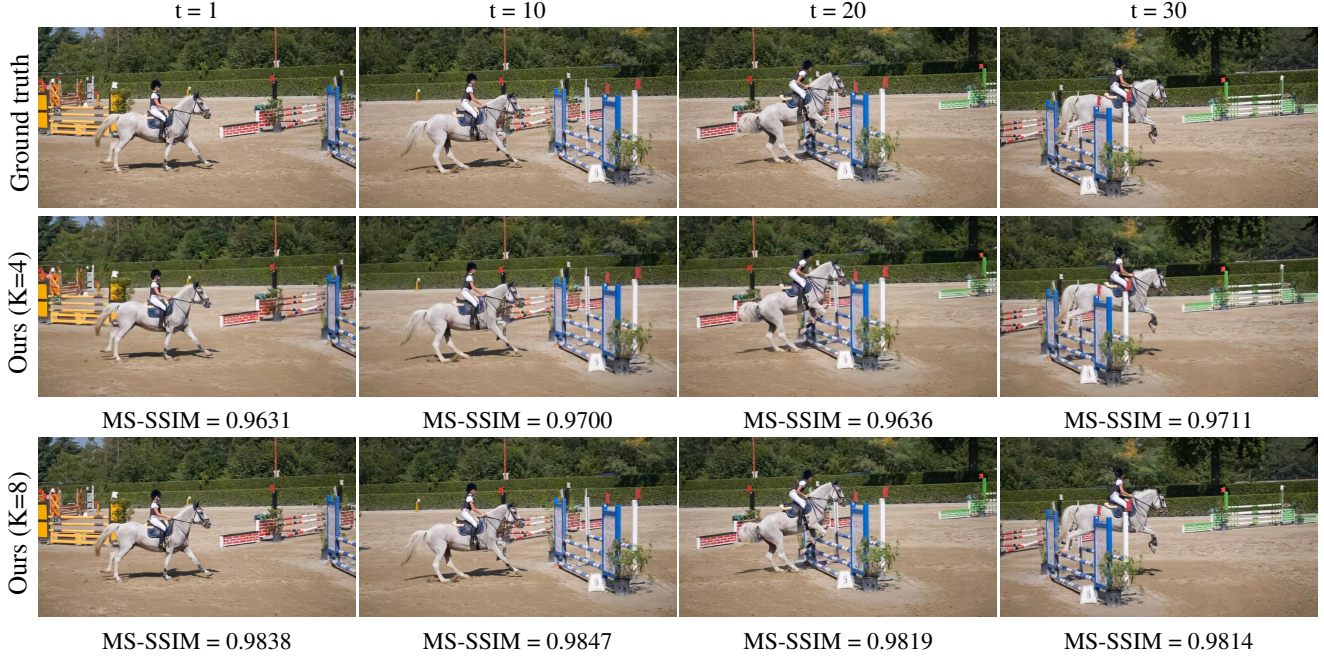


Figure 7: Our reconstructed high-resolution 480×848 video with two different compression parameters: $K = 4, 8$. A one-second video can be embedded in a 10-second audio signal and has high visual fidelity, with MS-SSIM beyond 0.96 for $K = 4$. When K is increased to 8, the visual fidelity is even higher but requires twice the number of audio samples for embedding.

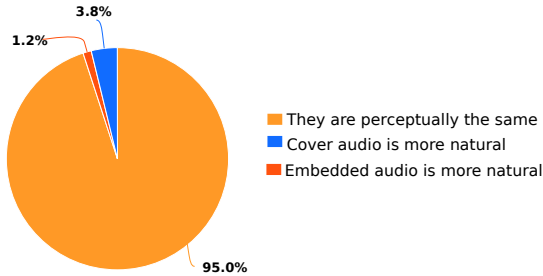


Figure 8: User study results on comparing the encoded audio and the original cover audio.

video. Under our steganography scheme, one-second 480p video can be embedded in 10-second audio with MS-SSIM beyond 0.96. In principle, our model can embed video of any resolution with high reconstruction fidelity.

5.5. Evaluation of Audio Fidelity

We conduct a user study comparing the encoded audio and the original cover audio with 30 participants. We randomly sample 10 encoded audio clips embedded with the DAVIS videos. In the user study, each participant hears the encoded audio and the original cover audio in random order once. Then the participant needs to choose one of the three options: 1) Audio A is more natural, 2) Audio B is more

natural, 3) They are perceptually the same. As shown in Figure 8, in 95.0% of the comparisons, participants cannot tell the difference. This user study shows that our model is perceptually transparent. (No claims on transparency to steganalysis are made.)

6. Discussion

We have proposed a flow-based model for hiding video in audio by combining image compression networks and invertible generative models. To enhance performance, we developed a novel optimized strategy for conversion between binary codes and floating-point latent variables. Our experiments have shown that our steganography scheme can efficiently conceal video in audio with high fidelity of reconstructed video and perceptually indistinguishable cover audio. We hope that our work can inspire more research on cross-modal steganography. Future work may investigate more robust steganography schemes that are resistant to data corruption or compression.

References

- [1] Muhammad Asad, Junaid Gilani, and Adnan Khalid. An enhanced least significant bit modification technique for audio steganography. In *International Conference on Computer Networks and Information Technology*, 2011. 2
- [2] Pooja P Balgurgi and Sonal K Jagtap. Intelligent processing: An approach of audio steganography. In *International*

- Conference on Communication, Information & Computing Technology (ICCICT)*, 2012. 2
- [3] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In *Advances in Neural Information Processing Systems*, 2017. 1, 2, 6
- [4] Walter Bender, Daniel Gruhl, Norishige Morimoto, and Anthony Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3.4):313–336, 1996. 2
- [5] Moustapha M Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *Advances in Neural Information Processing Systems*, 2017. 2
- [6] Nedeljko Cvejic and Tapio Seppanen. Increasing the capacity of lsb-based audio steganography. In *IEEE Workshop on Multimedia Signal Processing*, 2002. 2
- [7] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv:1410.8516*, 2014. 2
- [8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv:1605.08803*, 2016. 2
- [9] Fatiha Djebbar, Baghdad Ayad, Habib Hamam, and Karim Abed-Meraim. A view on latest audio steganography techniques. In *International Conference on Innovations in Information Technology*, 2011. 2
- [10] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv:1802.04208*, 2018. 2
- [11] Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting LSB steganography in color, and gray-scale images. *IEEE Multimedia*, 8(4):22–28, 2001. 2
- [12] Litao Gang, Ali N Akansu, and Mahalingam Ramkumar. Mp3 resistant oblivious steganography. In *ICASSP*, 2001. 2
- [13] Hamzeh Ghasemzadeh and Mohammad H Kayvanrad. Comprehensive review of audio steganalysis methods. *IET Signal Processing*, 12(6):673–687, 2018. 2
- [14] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, 1991. 4
- [15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014. 2
- [16] Daniel Gruhl, Anthony Lu, and Walter Bender. Echo hiding. In *International Workshop on Information Hiding*, 1996. 2
- [17] Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. In *Advances in Neural Information Processing Systems*, 2017. 2
- [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 2
- [19] Vojtěch Holub and Jessica Fridrich. Designing steganographic distortion using directional filters. In *IEEE International workshop on information forensics and security (WIFS)*, 2012. 2
- [20] Keith Ito. The LJ speech dataset. <https://keithito.com/LJ-Speech-Dataset/>, 2017. 5, 6, 7
- [21] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, 2018. 2
- [22] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014. 2
- [23] Felix Kreuk, Yossi Adi, Moustapha Cisse, and Joseph Keshet. Fooling end-to-end speaker verification with adversarial examples. In *ICASSP*, 2018. 2
- [24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5, 6
- [25] Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, 2005. 1, 2
- [26] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv:1609.03499*, 2016. 2
- [27] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel WaveNet: Fast high-fidelity speech synthesis. *arXiv:1711.10433*, 2017. 2
- [28] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc J. Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 6, 7
- [29] Fabien AP Petitcolas, Ross J Anderson, and Markus G Kuhn. Information hiding-a survey. *Proceedings of the IEEE*, 87(7):1062–1078, 1999. 1, 2
- [30] Tomáš Pevný, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In *International Workshop on Information Hiding*, 2010. 2
- [31] Lionel Pibre, Jérôme Pasquet, Dino Ienco, and Marc Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch. *Electronic Imaging*, 2016(8), 2016. 2
- [32] Wei Ping, Kainan Peng, and Jitong Chen. ClariNet: Parallel wave generation in end-to-end text-to-speech. *arXiv:1807.07281*, 2018. 2
- [33] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A flow-based generative network for speech synthesis. *arXiv:1811.00002*, 2018. 1, 2, 3
- [34] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics*, 2015. 2
- [35] Abdelfatah A Tamimi, Ayman M Abdalla, and Omaira Al-Allaf. Hiding an image inside another image using variable-rate steganography. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(10), 2013. 2
- [36] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *CVPR*, 2017. 2, 4, 6

- [37] Yuntao Wang, Kun Yang, Xiaowei Yi, Xianfeng Zhao, and Zhoujun Xu. CNN-based steganalysis of MP3 steganography in the entropy code domain. In *ACM Workshop on Information Hiding and Multimedia Security*, 2018. 1
- [38] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, 2003. 5
- [39] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2
- [40] Mazdak Zamani, Azizah Abdul Manaf, Rabiah Ahmad, Akram M Zeki, and Shahidan Abdullah. A genetic-algorithm-based approach for audio steganography. *World Academy of Science, Engineering and Technology*, 54:360–363, 2009. 2
- [41] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *ECCV*, 2018. 1, 2