Universiteit Leiden
The Netherlands

Bachelor Data Science and Artificial Intelligence

Video Steganography with Deep Neural Networks

Erki Elbrecht

Supervisors:
Dr. H. R. Doughty
Luc Sträter

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/07/2025

Abstract

The growing capabilities of generative video models have increased the need for reliable methods to distinguish authentic content from generated videos. Video watermarking, a method for hiding data in the video signal, offers a way to address this challenge. This thesis proposes a lightweight deep learning based video watermarking model that utilises multi-frame embedding to improve robustness, imperceptibility, and efficiency trade-off compared to existing approaches. Unlike prior publicly available work that embeds full watermark messages in every video frame or over the whole video, the proposed method distributes information across a small set of frames. A comprehensive set of video transformations, which include novel overlays and multiple compression algorithms are introduced during training to simulate real-world distortions. Results of the experiments demonstrate that the model achieves competitive robustness and imperceptibility compared to state-of-the-art baselines, while reducing computational cost. Furthermore, a real-world robustness evaluation using videos uploaded to and re-downloaded from a widely used video sharing platform reveals the limitations of current models for real-world use. The thesis highlights the effectiveness of temporal embedding and diverse training augmentations in the field of video steganography, while acknowledging the need for further studies.

Contents

1 Introduction

Video watermarking, a branch within the field of steganography, is the practice of hiding information within a video signal. The field has seen a significant amount of research in recent years, indicating its growing importance [AA24]. This growth can be tied to the arrival of many powerful generative video models, whose videos can be difficult to distinguish from real videos, even for a trained eye [VNS24] [Zha22]. The recently released Google Veo 3 [Dee24] and the well-known OpenAI Sora [LZL+24] are two examples of such models. These models also raise concerns about the possibility of generating misinformation, as generally, videos are seen as a more trustworthy medium than text [WTBR21] [SMC21]. Therefore, there is an urgent need to make it easier for humans to distinguish between generated and real content.

1.1 Motivation

While video watermarking is a valuable tool which could help address this issue, it is not a stand-alone solution. Watermarking would need to be deployed in tandem with well-formed policies for it to mitigate the spread of misinformation [HF24]. While the details of these policies are beyond the scope of this thesis, there have been proposals and efforts from various institutions [ZGC+24] [RvDK25] . Therefore, three possible applications of video watermarking are presented to show the importance and relevance of the work:

- Watermarking an AI generated video as the last part of the generation process ensures that the generated video can be more reliably classified as AI generated. This has already been proposed by some regulators [SB24] [HZL+25].
- Watermarking real videos with cryptographic signatures can help to verify that a video did indeed originate from a reliable source. This could be done even on a video recording device level [LWW+24].
- Moderators of a social media website can watermark an AI generated video to more effectively limit its spread and allow for independent parties to check if a video has been flagged [MKR+22].

1.2 Watermarking Criteria

The benefits of video watermarking are therefore evident, however, the question is how it could be achieved in a reliable manner. Three key aspects are used to evaluate how well information is hidden: robustness, imperceptibility and capacity. Firstly, the information must be hidden in a way such that it is hard to remove, both intentionally and unintentionally. This is known as the robustness of a watermark [ZNSL23] [KMP+22]. For example, information can be hidden in the metadata of a file, or in specific pixel values. However, this has limited robustness to unintentional attacks and even less for intentional ones. For instance, social media sites compress videos and remove unnecessary data, to reduce storage and streaming bandwidth requirements [YWZ+24].

Secondly, the imperceptibility of a watermark can be examined. This aspect of steganography examines how easy it is to tell that a video has been watermarked [WWW23]. For instance, the previous example of hiding information in file metadata would be imperceptible while watching the video, however, a large text overlay on the whole video would be considered perceptible. Imperceptibility is an important aspect to maintain the integrity of the original content [HWZ+22].

Thirdly, a watermark should be able to store as much information as possible [KP22]. In the context of stopping the spread of misinformation, the watermark should be able to contain information about a video, such as creation time, location data and a cryptographic signature among many other details. Thus, a watermark should also have a large enough capacity.

There is a trade-off between all three of these aspects. An imperceptible watermark is typically more fragile than a perceptible one, which means that the watermark has a lower robustness [AA24]. Whereas a high capacity watermark could be more perceptible as it allows for more information to be stored within the video file. For example, a logo in the corner of a video is less noticeable than multiple lines of overlaid text on a video. It is therefore important to balance all three aspects in a well-functioning steganography technique. In addition to these established criteria, there is a fourth aspect not inherent to steganography or watermarking, that is as integral as the other aspects for this thesis. This aspect is the efficiency of a watermarking technique, which is crucial for being able to process many high resolution videos at scale [ZWH+24].

1.3 Approach

This thesis proposes a video watermarking model using deep neural networks to balance watermark robustness, imperceptibility, capacity and efficiency. Moreover, this work aims to build upon the contributions in the paper Video Seal - a machine learning based video watermarking model by Meta [FEYM24], and DVMark - a model by Google [LLC+23], among others. Using neural networks to enhance video and image watermarking efforts has been a well researched topic, as demonstrated in multiple recent works [KP22] [KPP22] [LRD21]. One of the breakthrough papers within this topic is *HiDDeN: Hiding Data with Deep Networks* [ZKJFF18]. The benefits of this approach include a more context aware watermark, which helps to hide data in the areas of an image where it is less perceptible, and more robust watermarking, as it is difficult to account for many possible attacks on the watermark by traditional techniques [LLC+23].

The main contribution of the approach proposed in this thesis is how the temporal dimension of videos is utilized to embed a watermark in a video. Some publicly available models, like the aforementioned Video Seal, work by embedding all of the data into every frame of the video. While this simplifies training and could be more efficient, it does not use the temporal dimension of videos which could be utilized for better robustness. Other approaches, like DVMark, spread out the data over the entire video or a large number of frames. While this can improve robustness, it may introduce computational overhead, particularly in long high resolution videos. The goal of this approach is to create a more lightweight model that spreads out the data over a small number of video frames while maintaining comparable data capacity and without sacrificing robustness. This is done by adapting the original Video Seal model to use the temporal dimension of the video, while also reducing the number of layers. Furthermore, compared to previous works, an extended

amount of video transformations is implemented. These transformations include video overlays and more compression algorithms, which are applied during training to increase the robustness of the watermark to more real-world distortions. Lastly, the thesis contributes to the field by conducting a pilot study on how well the watermarking models perform in real-world scenarios, to see how the techniques can be improved in the future. This leads to the main research question of this thesis:

How do overlay transformations, compression techniques, and multi-frame embedding affect the robustness, imperceptibility, and efficiency of deep learning based video steganography models?

Two sub-questions were also identified:

- *How do overlay transformations and compression techniques impact the robustness and imperceptibility trade-off of the model and performance in real-world scenarios?*
- *How does utilizing the temporal dimension by spreading the watermark across multiple frames impact the trade-off between robustness, imperceptibility, and efficiency in a watermarking model?*

To answer these questions, a lightweight watermarking model will be developed and trained. The thesis will start by highlighting the important definitions, after which related work and the history of watermarking and steganography will be discussed. In the method section, the architecture, training pipeline and how the model works for non-training data videos will be discussed. The thesis goes on to detail how the model was trained and compares the results of the training with other models. Furthermore, an experiment simulating real-world scenarios will be conducted and results discussed. Finally, the conclusions drawn from the experiments will be presented, addressing the research questions directly. Additionally, potential avenues for further research will be discussed.

The author of this thesis acknowledges the support of the Leiden Institute of Advanced Computer Science and Dr. H. R. Doughty, whose supervision was instrumental to the development of this thesis.

2 Background

2.1 Traditional Video Watermarking

Video watermarking was first developed as a means to enforce copyright protection. The development of watermarking techniques began in the late 1990s with the growth of digital multimedia [DD03]. Basic techniques of video watermarking include hiding data in the least significant bits of each colour channel or in the discrete cosine transform of a frame - approaches that also apply to images. Traditional video-specific watermarking usually takes advantage of compression algorithms such as H.264. These methods work by manipulating motion vectors or exploiting Reversible Variable Length Codes (RVLC), which are used in video encoding to represent frequently appearing symbols [AP17]. While these approaches can achieve robust and imperceptible results, as was

demonstrated in the paper *Robust video watermarking of H.264/AVC* [ZHQM07], the watermark demonstrates robustness only within the constraints of the original encoding scheme. Re-encoding or stronger transformations can quickly reduce the accuracy of an extracted watermark. As traditional watermarking methods faced challenges in both robustness and adaptability, deep learning-based solutions started being explored.

2.2 Deep learning Based Approaches

Deep learning based video watermarking seeks to further increase the robustness of the watermark by not necessarily relying on compression algorithms. Most deep learning based watermarking models have a similar structure: an embedder, which takes an image or a set of video frames as an input to produce watermarked video frames, an extractor, which extracts the watermarked data, and some form of adversary to improve imperceptibility of the generated watermarks. While video-specific watermarking models appeared later, a breakthrough paper, *HiDDeN: Hiding data with deep networks*, was published in 2018, which demonstrated the viability of using deep neural networks for image watermarking [ZKJFF18]. The model achieved satisfactory results both in robustness and imperceptibility, compared to traditional methods. However, when directly applied to video frames, noticeable flickering artifacts were introduced. First video-based deep neural network watermarking models include RivaGAN [ZXCIV19] and VStegNet [MKNI19], both of which aimed to exploit the temporal dimension of videos by processing video frames simultaneously. Both models demonstrated strong performances, with VStegNet focusing efforts on watermark capacity and RivaGAN focusing on robustness by utilizing a novel adversarial network. However, both of the models need to process many frames in parallel, making these approaches inefficient for long high resolution videos.

2.3 Recent Advances

Newer video watermarking models include Video Seal [FEYM24], DVMark [LLC+23] and ItoV [YGW+23]. DVMark utilizes a novel multi-scale design where the watermarks are distributed across multiple spatial-temporal scales, which translates to improved robustness compared to non multi-scale models of the time. However, as pointed out in the paper for Video Seal, by processing videos in high resolution and multiple frames in parallel, the efficiency and usability of the model are questionable. ItoV, while still processing the entire video in parallel, improved on the efficiency of the model by merging the channel and temporal dimensions, allowing videos to be processed by 2D convolutions instead of 3D convolutional layers. Video Seal focuses more on the efficiency of the model, compared to previous works. Efficiency is achieved by disregarding the temporal dimension of the video and rather training a more powerful image watermarking model. A watermark is only generated at a configurable interval and the result is copied to the adjacent frames. While the approach achieves notable results, by not utilizing the temporal dimension, the potential maximum capacity of the model is brought into question. Furthermore, the technique can produce more noticeable artefacts for videos with lower frame rates or quicker camera movements.

2.4 Limitations

Despite significant progress in deep learning based video watermarking, there still exists a significant trade-off between imperceptibility, robustness, capacity and efficiency in all of the methods. Models that prioritize robustness often exhibit reduced imperceptibility, while the more lightweight models might have lower capacity or more significant watermarking artefacts. Furthermore, many previous methods rely on processing many frames or entire videos in parallel which can reduce efficiency and usability for real-world applications. These limitations motivate the model introduced in section 4. A hybrid watermarking model is introduced, which aims to balance these trade-offs, to create a smaller and more efficient model with similar capacity and robustness to previous larger models.

3 Definitions

This section aims to give an overview of the important key topics to contextualize the architectural components of the watermarking model introduced in the Section 4 of this thesis.

3.1 Convolutional Architectures for Image Processing

Image processing in neural networks is commonly performed using Convolutional Neural Networks (CNNs), which are a class of neural networks that utilize convolutional layers to extract features from images [LBBH02]. CNNs can be further categorized based on dimensionality - a 2D CNN processes images, while a 3D CNN can process videos or 3D data. Therefore, CNNs are crucial for implementing a neural network model for watermarking.

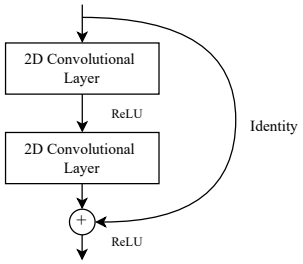


Figure 1: A 2D residual block with ReLU activation function

To improve the performance of CNNs and enable deeper architectures, convolutional layers can be arranged into residual blocks, as was introduced in the ResNet architecture [HZRS16]. A typical residual block includes two convolutional layers with an activation function and a skip connection, which helps to mitigate the vanishing gradient problem. An example of this block can be seen in Figure 1.

U-Nets are a type of convolutional neural network architecture originally proposed for biomedical image segmentation [RFB15]. They are composed of two parts, an encoder and a decoder. The encoder reduces the spatial dimensions of the input while increasing feature abstraction, helping the network capture high-level information. The decoder is concerned with the reverse process - the number of features are reduced to reconstruct the spatial dimension until the original dimensions have been achieved. Furthermore, the encoder layers are connected to decoder layers with the same spatial dimension by skip connections. These connections allow the network to preserve spatial information. This configuration enables the network to combine high-level semantic understanding with spatial detail, making it well-suited for segmentation or watermarking. A diagram of this can be seen in Figure 2.

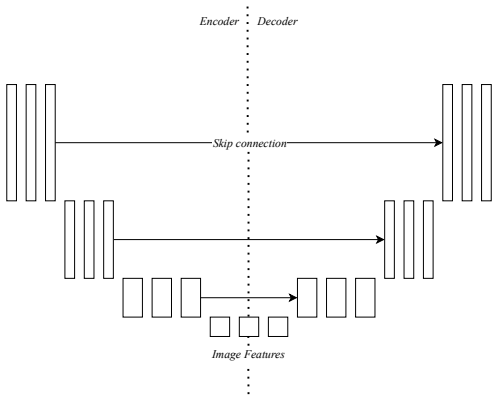


Figure 2: The U-Net architecture

3.2 Transformer Architectures for Vision Tasks

While convolutional neural networks have been the leading architecture for image processing tasks, recent advancements have shown that transformer architectures, which were originally developed for natural language processing [VSP+17], can also be effective for visual data. Transformers capture global dependencies through self-attention layers, allowing them to capture long-range dependencies across an image. This global context awareness enables transformers to perform well in tasks where spatial relationships and overall structure are important. This technique was successfully used on vision tasks in the Vision Transformer model or ViT, which showed that a transformer-only architecture can achieve similar results as CNNs [DBK+20]. A diagram of a transformer encoder is shown in Figure 3. In the context of watermarking, these properties may improve the robustness and accuracy of message extraction, especially when information is distributed across multiple regions or frames.

4 Method

This thesis aims to balance the previous temporal information integrating approaches and Video Seal by distributing the watermark over a small number of frames instead of embedding the full

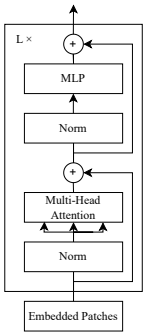


Figure 3: A diagram of a transformer encoder, where L is the number of layers

binary message on a single frame. Furthermore, to improve efficiency, a watermark propagation technique is introduced that enables robust watermarking without generating a watermark for every frame. Similarly to Video Seal [FEYM24] this approach utilizes a ResNet-based U-Net for the embedder, although with 3D convolutional layers, and a Vision Transformer for the extractor [DBK+20].

At a high level, the model receives a set of frames from the video being watermarked and a binary message to be embedded. The frames are then encoded into a latent representation and combined with an embedding of the binary message. The features are merged and decoded into a watermark, which is subsequently blended with the original frames of the video. To recover the original message, frames are divided into image patches which are encoded into features. A weighted pooling mechanism aggregates the information from multiple frames and patches to reconstruct the hidden message. The model can be applied to videos of arbitrary length and resolution. It is implemented in PyTorch, and the source code and weights are publicly available.

4.1 Architecture

The architecture of the proposed watermarking model can be divided into two main components: an embedder, which is responsible for generating the watermarks, and an extractor, which extracts a hidden message from a watermarked video. A high-level diagram of this can be seen in Figure 4. This thesis will go on to specify each component in further detail.

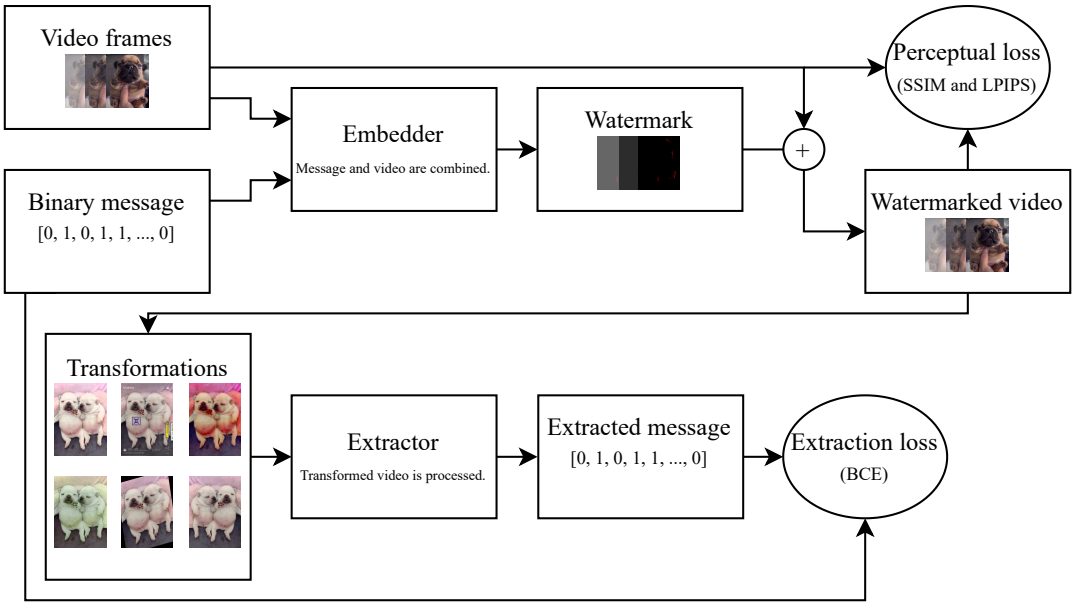


Figure 4: A diagram depicting a high-level overview of the model.

4.1.1 Embedder

The embedder consists of a 3D ResNet-based U-Net and a message embedding part. The input to the embedder is F frames of video $x \in \mathbb{R}^{(3\times F\times H\times W)}$ and a binary message $m \in \{0,1\}^{n_{\text{bits}}}$ where n_{bits} is the length of the message. The output of the embedder is a watermark $w \in \mathbb{R}^{(3\times F\times 256\times 256)}$ that can be additively blended with the original video. While the number of frames F and the message length n can be chosen arbitrarily, for this model, $F = 3$ and $n_{\text{bits}} = 96$ was chosen. This amount of bits was chosen because other models, such as Video Seal [FEYM24], are also trained on 96 bits, making comparisons easier. The amount of frames was chosen because the model does not reduce the size of the temporal dimension during the encoding part of the U-Net, which means the frame count should remain low. Furthermore, a lower number of frames ensures that the training can be successfully completed within the time frame of the thesis. Table 1, demonstrates the structure of the embedder in further detail.

The U-Net is structurally similar to the one used in Video Seal and TrustMark [BAC23]. However, instead of 2D convolutional layers, 3D ones are used instead to account for the temporal dimension. The U-Net is further made up of a decoder, which uses upscaling blocks, and an encoder, which uses downscaling blocks. The blocks are connected with skip connections. Downscaling blocks or DownBlocks, consist of a bilinear downscaling layer, which only downscales the last two spatial dimensions, and a 3D ResNet block. Similarly, upscaling blocks or UpBlocks consist of a bilinear upscaling layer and a 3D ResNet block. ResNet blocks are made from two convolutional layers with ReLU and batch normalization. A skip connection is implemented with a convolutional layer, having a kernel size of 1.

First, the video frames are resized to 256 by 256 and go through an initial 3D ResNet block. Three DownBlocks encode the frames into d_{vid} feature maps, to which the message embedding is concatenated. The message embedding is constructed from a binary message lookup table $T \in \mathbb{R}^{(n \times 2 \times d_{\text{msg}})}$. This means that the table contains unique features for every bit position and for each of the two states. The features for each bit are added together to achieve one representation $r \in \mathbb{R}^{d_{\text{msg}}}$ for the whole binary message. Finally, the message embedding is repeated so that it matches the size of the feature maps for the video.

The concatenated features are then merged by three bottleneck 3D ResNet blocks, which reduce the amount of channels back to d_{vid}. Next, three UpBlocks with skip connections reduce the number of channels and increase the spatial dimensions. A final 3D ResNet block decodes the features back into video frames, which now represent the watermarks. These watermarks are multiplied by a watermark strength constant α and added to the original frames to produce watermarked frames. For this embedder, $d_{\text{msg}} = 192$ and $d_{\text{msg}} = 128$ was chosen as Video Seal, which this embedder is based on, has shown success using these parameters.

Encoder	**Decoder & Embedding**
$x \in \mathbb{R}^{(3 \times 3 \times H \times W)}$	$m \in \{0,1\}^{n_{\text{bits}}}$
Resize $\rightarrow \mathbb{R}^{(3 \times 3 \times 256 \times 256)}$	Embedding, Repeat $\rightarrow m \in \mathbb{R}^{(d_{\text{msg}} \times 3 \times 32 \times 32)}$
3DResNetBlock $\rightarrow \mathbb{R}^{(d_{\text{vid}}/8 \times 3 \times 256 \times 256)}$	Concat $\rightarrow \mathbb{R}^{(d_{\text{msg}}+d_{\text{vid}} \times 3 \times 32 \times 32)}$
3×DownBlock $\rightarrow \mathbb{R}^{(d_{\text{vid}} \times 3 \times 32 \times 32)}$	6×Bottleneck $\rightarrow \mathbb{R}^{(d_{\text{vid}} \times 3 \times 32 \times 32)}$
	3×UpBlock $\rightarrow \mathbb{R}^{(d_{\text{vid}}/8 \times 3 \times 256 \times 256)}$
	3DResNetBlock $\rightarrow \mathbb{R}^{(3 \times 3 \times 256 \times 256)}$

Table 1: Architecture of the embedder.

4.1.2 Extractor

The input for the extractor are the watermarked video frames $x \in \mathbb{R}^{(3 \times F \times 256 \times 256)}$. The output is a logit representation of the binary message $m \in \{0,1\}^{n_{\text{bits}}}$ hidden in the watermarked video. The extractor consists of four main parts: a patch embedding module which divides the video frames into 16 by 16 pixel patches over a token embedding dimension d_{model}, a vision transformer [DBK+20][LMGH22], which processes the patch tokens into d_{model} features, a convolutional neck, which projects features into a lower-dimensional space and fuses the features temporally, and a final weighted pooling layer which combines all patch tokens in one video to produce a logit representation of the binary message. While this implementation is similar to the work done in Video Seal, it instead uses a smaller number of transformer blocks, uses 3D convolutions in the neck and uses weighted pooling instead of mean pooling. Table 2, shows the architecture of the extractor.

First, the temporal and batch dimension are merged, as all frames are first processed individually. Next, all frames of the video are divided into 16 by 16 d_{model} dimensional tokens with a 2D convolutional layer. Learned positional embeddings are added to the tokens. These tokens are passed to L transformer encoder blocks. The last part of the patch extractor is used to normalize the encoded patches.

The convolutional neck receives the encoded patches and convolves over the batches, combining the temporal dimension and patch embedding dimensions. Weighted pooling is applied to the tokens to produce d_{model} features for a video. A final feed-forward layer produces a logit representation of the binary message. For the extractor, $d_{\text{model}} = 384$ and $L = 6$ was chosen as a reasonable balance between expressiveness and efficiency.

Patch Extractor ($BF = B \cdot F$)
$x \in \mathbb{R}^{(B \times 3 \times F \times 256 \times 256)}$
Reshape $\rightarrow \mathbb{R}^{(BF \times 3 \times 256 \times 256)}$
PatchEmbed $\rightarrow \mathbb{R}^{(BF \times 16 \times 16 \times 384)}$
Positional Embedding $\rightarrow \mathbb{R}^{(BF \times 16 \times 16 \times 384)}$
$L\times$ ViTDet Block $\rightarrow \mathbb{R}^{(BF \times 16 \times 16 \times 384)}$
Reshape $\rightarrow \mathbb{R}^{(B \times 384 \times F \times 16 \times 16)}$
Neck (Conv3D/LayerNorm) $\rightarrow \mathbb{R}^{(B \times 256 \times F \times 16 \times 16)}$
Flatten + Permute $\rightarrow \mathbb{R}^{(B \times (F \cdot 256) \times 256)}$
Weighted Pooling $\rightarrow \mathbb{R}^{(B \times 256)}$
Linear $\rightarrow \mathbb{R}^{(B \times n_{\text{bits}})}$

Table 2: Architecture of the extractor.

4.2 Training Pipeline

The model is trained end-to-end, both the extractor and the embedder are trained at the same time. A diagram of how all components are combined can be seen in Figure 4. Since the training involves multiple objective functions and complex transformations, the training of the model is divided into multiple parts, which will be discussed in section 5.1.

4.2.1 Training Objectives

The training objectives or losses are divided into two categories: perceptual losses and extraction loss. The perceptual losses are responsible for the imperceptibility of the watermark, while the extraction loss is responsible for the robustness or the accuracy of the extraction. While many previous models such as DVMark [LLC+23] or Video Seal [FEYM24] also include an adversarial loss, this model omits the use of this loss in favour of more perceptual losses. The reason for this is that adversarial training comes with a computational overhead, which could not be afforded in the training of this model.

The perceptual losses chosen for this model are Learned Perceptual Image Patch Similarity (LPIPS) [ZIE+18] and Structural Similarity Index (SSIM) [WBSS04]. These two metrics were selected due to their complementary properties in capturing perceptual quality:

- LPIPS compares feature representations of given images from pre-trained convolutional networks. This can align well with human perception of image similarity. It has been shown to perform better than pixel-wise losses in measuring perceptual closeness.

- SSIM, on the other hand, shows the structural similarities between images, providing a more robust metric, where pre-trained networks might overlook details.

SSIM is implemented in PyTorch, while LPIPS is provided in a python package with the same name. The VGG based back-end neural network for LPIPS was used for faster and more stable training.

The extraction loss for the watermark is Binary Cross Entropy (BCE), as the problem of predicting a binary message is a binary classification problem. It penalizes wrong bit predictions and provides a stable gradient signal during training. Furthermore, BCE loss is commonly used in prior work such as in Video Seal and TrustMark [BAC23].

To optimize the training process, the losses are combined using both adaptive weights and non-adaptive weights. The adaptive weights scale the loss by the norm of its gradient, which is an approach also utilized in Video Seal [FEYM24]. This approach addresses the gradient scale imbalance between the losses. The non-adaptive weights are set manually, but can be adjusted during the training to direct the goals of the model. The values for the non-adaptive weights are discussed in section 5.1. The full loss of the model is then defined by the following equation:

$$L = \lambda_{\text{BCE}} * \tilde{\lambda}_{\text{BCE}} * l_{\text{BCE}} + \lambda_{\text{LPIPS}} * \tilde{\lambda}_{\text{LPIPS}} * l_{\text{LPIPS}} + \lambda_{\text{SSIM}} * \tilde{\lambda}_{\text{SSIM}} * l_{\text{SSIM}}$$

Where λ refers to non-adaptive weights, $\tilde{\lambda}$ refers to adaptive weights and l refers to a loss value.

4.2.2 Transformations

Transformations take place between the embedder and the extractor. They are important for the robustness of the model, as the goal is to simulate both intentional and unintentional attacks on the watermark. For the network to be fully end-to-end trainable, the transformations must be differentiable, which is not the case for many available implementations. Therefore, multiple transformations have been implemented from scratch for differentiability and speed. These are categorized into three groups: valuemetric, which change the pixel values, geometric, which modifies the geometry of a frame, and overlay, which adds overlays to a frame. All the implemented transformations can be seen in Table 3.

It is possible to further separate the transformations into per-frame and per-video transformations. In the model, per-frame transformations mean that all the frames in a video are modified the same way and with the same parameters, whereas per-video transformations utilize inter-frame information, which affects different frames in distinct ways. For example, a per-frame brightness change translates to all the frames of the video being subjected to a brightness change, using the same parameters. However, a per-video compression transformation would introduce artifacts that are not uniform across multiple frames. For this model, the only per-video transformations are video compression algorithms. The rest of the transformations are applied uniformly across all frames.

Most video compression algorithms are non-differentiable, which means that gradients of the objective function can not be back-propagated to update the weights of the embedder. One way to

Transformation	Type	Parameters and choice for training	Probability
Brightness	Valuemetric	Brightness α, random uniform in range $0.8 - 1.2$	1.0
Contrast	Valuemetric	Contrast α, random uniform in range $0.8 - 1.2$	1.0
Saturation	Valuemetric	Saturation α, random uniform in range $0.8 - 1.2$	1.0
Hue	Valuemetric	Hue shift α, random uniform in range $-0.2 - 0.2$	1.0
Gaussian blur	Valuemetric	Kernel size k, random uniform odd in range $3 - 11$	0.2
JPEG Compression	Valuemetric	Quality q, random integer in uniform range $50 - 85$	0.8
Rotation	Geometric	Radians r, random uniform in range $-1.2 - 1.2$	0.4
Horizontal flip	Geometric	-	0.4
Crop	Geometric	Top t, left l, bottom b and right r percentages, all random uniform in range $0.1 - 0.2$	0.4
Perspective	Geometric	Horizontal shift h and vertical shift v, both random uniform in range $-0.2 - 0.2$	0.4
Sticker overlay	Overlay	Random sticker x, scale s, top t and left l percentages. Scale uniform in range $1.0 - 2.0$, top and left uniform in range $0.0 - 1.0$	0.6
Emoji overlay	Overlay	Random emoji x, scale s, top t and left l percentages. Scale uniform in range $1.0 - 2.0$, top and left uniform in range $0.0 - 1.0$	0.8
Frame overlay	Overlay	Random frame f	0.4
H.264 Compression	Valuemetric	Constant rate factor c in integer range $12 - 27$	$0.8 \times \frac{1}{3}$
AV1 Compression	Valuemetric	Constant rate factor c in integer range $12 - 27$	$0.8 \times \frac{1}{3}$
VP9 Compression	Valuemetric	Constant rate factor c in integer range $12 - 27$	$0.8 \times \frac{1}{3}$

Table 3: All transformations with the type and parameters used during the training of the model.

circumvent this limitation is to use an approximate differentiable version of a video compression algorithm instead [ZKJFF18]. However, implementing differentiable approximations for multiple algorithms is a monumental task. Therefore, another solution was chosen, similarly to Video Seal [FEYM24], where a gradient of a non-differentiable operation is approximated using the identity function [BLC13].

$$x_{transformed} = x_{identity} + nograd(\, T(\, x_{identity}\,) - x_{identity}\,)$$

Where T is the non-differentiable transformation function and *nograd* represents a function in which gradients are not propagated. The method, known as the straight-through estimator, allows gradients to bypass the non-differentiable transformation by treating it as the identity during back-propagation. While approximate, it has been shown to work well in tasks such as watermarking.

The per-frame transformations implemented are: brightness, contrast, saturation, hue, rotation, horizontal flip, crop, perspective, Gaussian blur, sticker overlay, emoji overlay, frame overlay and

JPEG compression. All valuemetric transformations take place in the HSV color space, which means there is a minimal approximation error between the non-transformed and transformed images. The per-video transformations are: H.264 compression [Ric11], AV1 compression [DRH18] and VP9 compression [MBG+13]. These compression transformations are implemented using PyAV, which is a python wrapper for the FFmpeg library. JPEG compression is implemented with the PyTurboJPEG package, which is a wrapper for the TurboJPEG library. All other transformations are implemented solely in PyTorch.

While valuemetric and geometric transformations are common in data augmentation for machine learning [MG18], the overlay transformations are a more novel approach created for the training of this model. The overlay transformations utilize a library of emojis, stickers and frames with transparent backgrounds. These overlays are then combined with the original video frame by alpha blending. Emojis and stickers may have various scales and positions, however, frames are always the full size of a video. The stickers differ from emojis by complexity and size - stickers are bigger and can exhibit more complex blending. These transformations were created to achieve greater robustness against possible attacks on the watermark. Examples are visible in Figure 5. The emojis were downloaded from OpenMoji [Ope24], the stickers were downloaded from various sources all with an MIT license, and the frames were manually designed in a vector graphics editor. The frames are available in the project repository.

(a) Frame transformation. (b) Sticker transformation. (c) All overlay transformations.

Figure 5: Examples of overlay transformations.

4.3 Inference

For real-world use, the model needs to be capable of handling longer than three frame sequences and high resolution videos. However, the model is trained at a fixed sequence length and video resolution - 3 and 256 by 256 respectively. Therefore, two methods are introduced to make it possible to watermark longer and larger videos with the model.

The first method addresses high resolution videos, which involves bilinearly scaling the watermark

to fit the size of full resolution video frames. This is defined by the following equation:

$$x_{watermarked} = x_{identity} + \alpha \cdot resize(y)$$

Where x is video frames, y is the watermarks and function $resize$ resizes the watermark y to the dimensions of x using bilinear interpolation. Furthermore, the watermark strength can be adjusted via a scaling factor $\alpha \in [0, 1]$. This method is also used in Video Seal and TrustMark [BAC23].

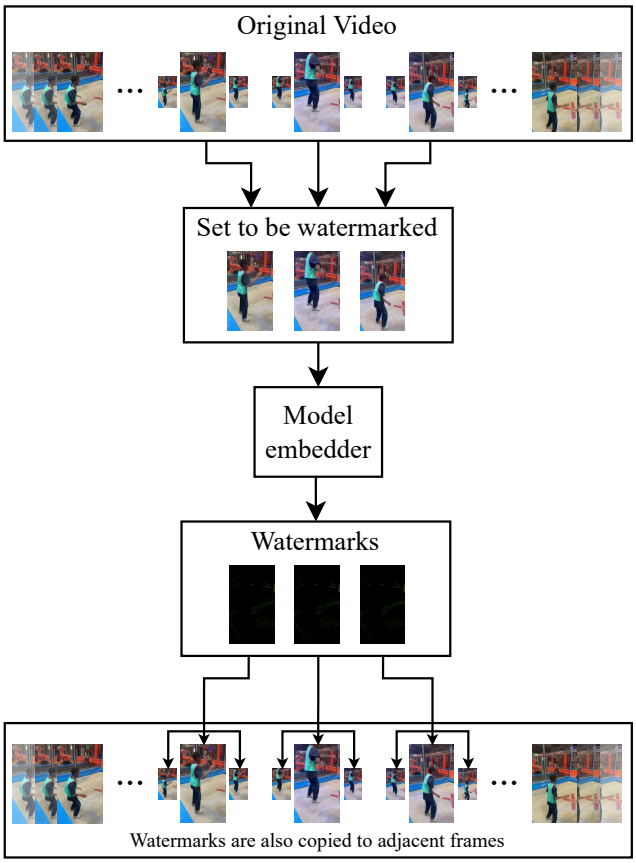


Figure 6: Depiction of how watermarks are propagated over a video sequence. $s = 1$

The second method addresses how videos longer than the amount of frames the model was trained on can be watermarked. For this model, the chosen frame amount f is 3. This means that the model only watermarks three frames of video at a time. While it is possible to process all frames in a video three frames at a time, this is inefficient. Inspiration is taken from the Video Seal model, where a watermark is only generated at an interval k frames, after generation, $k - 1$ further frames receive the same watermark. However, this method cannot be directly applied to this watermarking model as watermarks for multiple frames are generated in parallel. Therefore, a modified approach defines a constant $s \in \mathbb{N}$ indicating the spread of a watermark. The frames to be watermarked are chosen by selecting the frame after every $2s$ frames, except for the beginning of the video where a frame is selected after s frames. This method ensures that each selected frame is surrounded by s unselected frames in both temporal dimensions. After the watermarks are generated, three at a time, the

resulting watermark for every chosen frame is also copied to s adjacent ones. This approach is also visualised in Figure 6, where $s = 1$. The benefit of this approach is that a frame always receives the closest possible watermark, which means that the watermark is more imperceptible and temporally stable.

5 Experiments

To answer the research questions posed in Section 1, the model is trained and the results are compared against previous models. This section will also discuss the positive and negative aspects of the proposed approach. Furthermore, an experiment is conducted to evaluate the usability of the watermarking models for real-world scenarios.

5.1 Training

The model is trained on an Nvidia A100 GPU for 200 epochs with 1500 steps per epoch and a batch size of 16. The AdamW [LH17] optimiser is used along with a cosine scheduler, where the starting learning rate 1.0×10^{-5} is gradually reduced to zero.

The Segment Anything video dataset [RGH+24] is used for training. This dataset was chosen for its size, availability and diversity of real-world video content. During every step, three-frame sequences are randomly selected from the videos, which are downscaled to 256×256 resolution and normalized to the range $[-1, 1]$. Since the structure of the model relies on the temporal dimension of videos, there is no image pre-training as in some other models [FEYM24].

To make learning easier in early epochs, a training schedule is used to gradually introduce transformations to the model. The model starts learning without any transformations. On epoch 5, the valuemetric transformations and JPEG compression are introduced. On epoch 10, the geometric transformations are enabled. Overlay transformations are enabled on epoch 15, and video compression is introduced in epoch 30. Early transformations such as valuemetric and JPEG compression preserve the structure of the input to the extractor, making them easier to learn from. However, later transformations like geometric, overlays, and video compression significantly alter the videos, making early learning unstable, which is why these are introduced later. The transformations and their types are shown in Table 3.

The training objectives are not introduced all at once. During the first 100 epochs, the model optimises only BCE and MSE losses with the weights 1.0 and 0.1 respectively. This is also done to stabilize the training process in the early epochs as the other objectives, SSIM and LPIPS, have higher variance and less stable gradients. The SSIM and LPIPS are both introduced with a weight of 0.05 after the 60th epoch, while the other two losses retain their weights throughout the training process. Furthermore, the gradient norm based adaptive weights are also included after epoch 80, with the aim to further optimize for imperceptibility. The full loss equation and individual losses are discussed in Section 4.2.1.

The model is validated every 5 epochs on the validation split of the Segment Anything video dataset. The final model is chosen based on the lowest combined loss validation performance. This

training schedule, both in augmentations and objectives, was found to improve convergence speed and imperceptibility of the model. Graphs for the training process are shown in Appendix B.

5.2 Results

To gain insight into the performance of the model, it is compared against four other watermarking models: VideoSeal [FEYM24], CIN [MGH+22], HiDDeN [ZKJFF18] and MBRS [JFZ21]. The bit capacity and image processing size of the models is reported in Table 4. The aim is to compare and motivate performance in three key aspects: robustness, imperceptibility, and efficiency. Capacity, which was introduced in Section 1.2, is not included in the evaluation due to architectural constraints: each model has a fixed capacity, which is not adjustable without architectural modifications and retraining. A proper analysis of capacity trade-offs would require reimplementing each model across a range of capacities, which is an important direction for future work, but beyond the scope of this thesis.

The baseline models tested in this paper are all solely image-based models except for Video Seal, which is designed to watermark both images and videos. This is because of the availability of trained video watermarking models. For example, influential models like DVMark [LLC+23] and ItoV [YGW+23] do not have public code repositories or weights. However, the comparison between models is still relevant as a video can be watermarked by separately watermarking every frame.

	HiDDeN	CIN	MBRS	Video Seal	This model
Bit Capacity	48	30	256	96	96
Image Resolution	256×256	128×128	256×256	256×256	256×256

Table 4: Bit capacity and image resolution specifications for the compared watermarking models.

All key aspects are evaluated using the test split of the Segment Anything video dataset, however, the models were all trained on datasets used in their original papers. Furthermore, to account for the proposed model which spreads the watermark over three frames, and ensure a fairer comparison between the models, all models are required to watermark videos in three frame segments for which the reported metrics are averaged.

5.2.1 Robustness

Watermark robustness can be evaluated as the bitwise accuracy of the predicted message to the original message. However, since the models do not have the same bit capacity, the metric can be misleading as it is easier to reconstruct shorter messages. Therefore, to evaluate robustness relative to capacity, a p-value associated to a given bit accuracy is calculated, similarly to the analysis done in Video Seal [FEYM24]. Furthermore, as noted in the paper, the logarithm of the p-value can also be interpreted as the probability of observing a given bit accuracy or higher under the null hypothesis of a random guess. Lower $\log_{10}(p)$ values (e.g., below -15) indicate strong robustness.

The p-value normalizes accuracy by accounting for the difficulty of predicting longer messages. The formula is as follows:

$$\text{p-value}(m_{\text{pred}}, m) = \sum_{j=k}^{n_{\text{bits}}} \binom{n_{\text{bits}}}{j} \cdot \left(\frac{1}{2}\right)^{n_{\text{bits}}}$$

Where n_{bits} is the capacity of the model and k is defined as:

$$k = \lceil \text{bit\_accuracy}(m_{\text{pred}}, m) \cdot n_{\text{bits}} \rceil$$

And bit_accuracy is defined as:

$$\text{bit\_accuracy}(m_{\text{pred}}, m) = \frac{1}{n_{\text{bits}}} \sum_{i=1}^{n_{\text{bits}}} \mathbb{1}\left[m_{\text{pred}}^{(i)} = m^{(i)}\right]$$

Bitwise accuracy and its associated p-values are reported for each of the models over six experiments, testing different categories of transformations between watermarking and watermark extraction:

1. Identity / No transformations.
2. Valuemetric transformations with the probability of each transformation 1.0.
3. Geometric transformations with the probability of each transformation 1.0, except horizontal flip and Gaussian blur which both have a probability of 0.5.
4. Overlay transformations with the probability of sticker and emoji overlay 1.0 and frame overlay 0.5.
5. Compression transformations, with all probabilities 1.0.
6. All transformations combined with the aforementioned probabilities.

All transformations, their corresponding parameters and types are listed in Table 3. For all models and experiments, the order of videos to be watermarked, the seed for the random transformations, and the messages to be embedded are fixed. This makes for a more fair comparison. The results are reported in Table 5.

	HiDDeN		CIN		MBRS		Video Seal		This model	
	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$
Identity	1.00	−14.4	1.00	−9.0	1.00	−77.0	0.99	−28.9	0.99	−28.9
Valuemetric	0.92	−10.2	0.98	−9.0	0.99	−72.5	0.99	−28.9	0.98	−26.9
Geometric	0.50	−0.3	0.50	−0.2	0.50	−0.3	0.89	−15.8	0.86	−13.2
Overlay	0.75	−3.4	0.83	−3.8	0.86	−15.6	0.91	−17.7	0.98	−26.9
Compression	0.71	−2.9	0.72	−2.0	0.75	−9.0	0.85	−13.2	0.92	−18.8
Combined	0.50	−0.3	0.50	−0.2	0.50	−0.3	0.82	−10.3	0.84	−11.7

Table 5: Bitwise accuracy and associated p-values under various transformations.

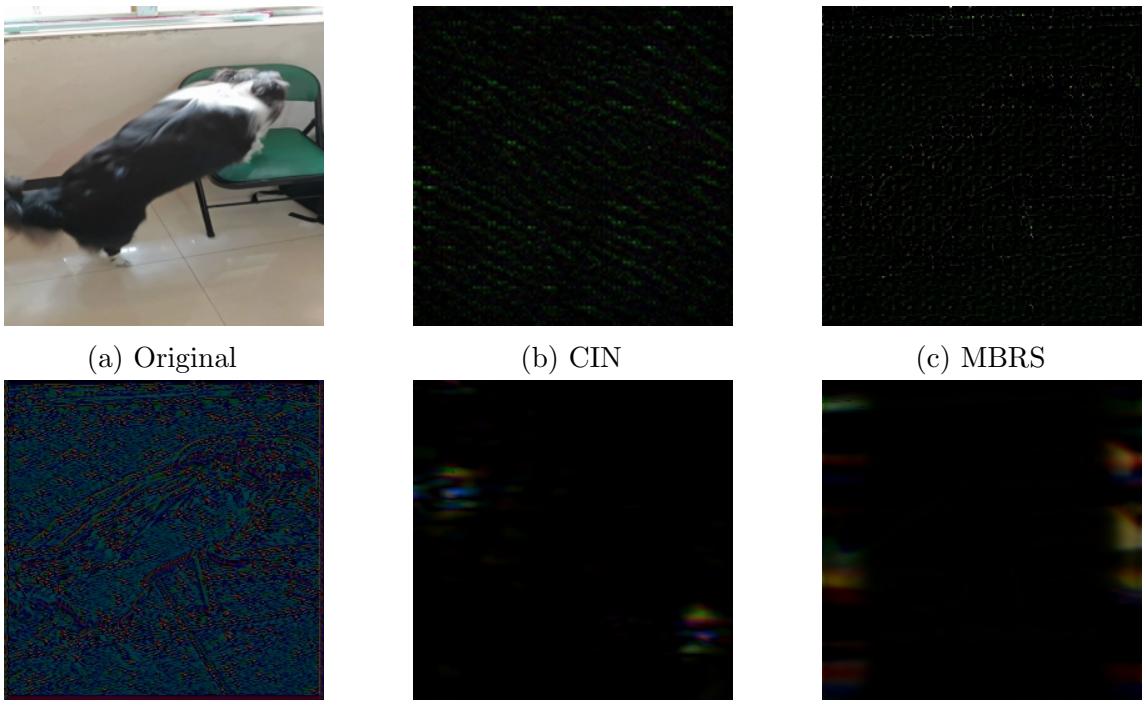
The achieved results for baselines in the experiments for identity, valuemetric and geometric are comparable to the reported results in the Video Seal paper with the exception of HiDDeN, where the authors reimplemented the model and trained it on extra transformations that were not in the original HiDDeN model. Bigger differences arise in the compression experiment, which could be due to the strength of the compressions and the extra compression algorithms implemented in this thesis. Furthermore, since the models HiDDeN, MBRS and CIN are trained with transformations that do not modify the geometry of the image, the models did not achieve notable results in the corresponding experiments with performance approximating that of random guessing.

Compared to Video Seal, which received the best results of all baseline models, the model implemented in this paper had comparable, but slightly inferior results in identity, valuemetric, and geometric transformation experiments. This could be attributed to the smaller size of the model and a shorter training process. However, in the overlay and compressions transformations experiments, Video Seal observed worse results. This is significant as it indicates that the extra information applied to frames and the loss of the original frame information in places where the overlays are applied, have an effect on the robustness of the Video Seal model. For the other models, overlay transformations also presented a challenge, reinforcing the importance of including such transformations in the training process.

Overall, in the combined transformations experiment, the proposed model achieved slightly higher performance in robustness than Video Seal and much greater performance than other baseline models. This indicates that spreading message information over multiple frames does improve the robustness of the model. However, since the proposed model is not as deep as the model proposed in Video Seal, the potential robustness gains of this technique could be even bigger.

5.2.2 Imperceptibility

To measure and compare the imperceptibility of the watermarks, three metrics are introduced. These are: Peak Signal to Noise Ratio (PSNR), SSIM [WBSS04], and LPIPS [ZIE+18]. The details of SSIM and LPIPS are described in Section 4.2.1 as these metrics are used as losses to train the proposed model. Higher SSIM (ranging from 0 to 1) and PSNR values indicate higher imperceptibility while lower LPIPS (also ranging from 0 to 1, with lower being better) values indicate lower perceptual difference. PSNR is a standard and widely adopted metric in image and video processing to assess the degradation between an original and a distorted signal. It provides a simple and interpretable measure of how much a watermarked frame deviates from the original. Typically, PSNR values above 30 dB indicate more imperceptible distortions to the human eye, while values below that may introduce visible artifacts. However, it must be noted that while all three metrics give a good indication about the imperceptibility of a watermark, the values do not always perfectly correlate with human perception. This is especially true for videos as some perturbations in the frames are more noticeable than others, while yielding similar scores in the introduced metrics. Therefore, samples of generated watermarks are displayed in Figure 7, to serve as an indication of how noticeable a watermark might be. Furthermore, the watermarks displayed in the figure have also been strengthened so that they are more visible. The results for the imperceptibility metrics averaged over the test split of the dataset are given in Table 6.



(a) Original (b) CIN (c) MBRS

(d) HiDDeN (e) Video Seal (f) Proposed model

Figure 7: Examples of generated watermarks for a video frame.

	HiDDeN	CIN	MBRS	Video Seal	This model
PSNR	32.18	42.32	43.66	46.86	38.79
SSIM	0.934	0.984	0.989	0.997	0.994
LPIPS	0.194	0.021	0.009	0.012	0.033

Table 6: Imperceptibility metrics for the tested models.

To evaluate the proposed models performance, the results are compared for every metric. Firstly, the proposed model achieves a PSNR of 38.79, which is higher than HiDDeN, but lower than the other baseline models. This suggests that while the model achieves reasonable visual fidelity, it introduces more signal level perturbations than the top baselines. However, it still exceeds the 35 dB commonly accepted threshold for imperceptibility. For SSIM, the proposed model achieves a score close to Video Seal, which performs the best out of all the baselines. This indicates that the structural information of the original frames is very well preserved. The proposed model places in the middle in terms of LPIPS, which shows that while the watermarked frames are perceptually close to the original, there may be more noticeable artifacts compared to leading baselines.

From visual inspection of Figure 7, it can be seen that the generated watermarks by the proposed model are similar to those generated by Video Seal - more washed out blobs instead of sharper artifacts produced by other baseline models. This may indicate that the model is more suitable for video watermarking than the image-based baselines, as smoother artifacts are less noticeable in motion. While the proposed model does not achieve state-of-the-art PSNR or LPIPS scores, the visual quality remains high and well above conventional thresholds.

5.2.3 Efficiency

To compare the relative efficiency of the models, the number of Floating Point Operations (FLOPs) is measured separately for both the extractors and the embedders. Because these operations dominate the computational cost in neural networks, FLOPs provide a hardware-agnostic measure of model efficiency. Fewer number of FLOPs generally implies faster inference times and reduced energy consumption, making it a useful metric for comparing model efficiency. For this experiment, FLOPs are measured per inference pass, excluding the computations for loss and back-propagation. The spatial dimension of the frames passed to the models is indicated in Table 4. Furthermore, the metric will be reported in Giga-FLOPs (GFLOPs) as the number of FLOPs for modern neural networks can be large. To make the comparison fair between the baselines, which can watermark one frame at a time, and the proposed model, which watermarks three frames at a time, the watermark propagation techniques need to be taken into account.

For the baselines, the propagation technique introduced in Video Seal [FEYM24] is used, which entails generating only one watermark per $k+1$ frames. More concretely, generating one watermark applies it to k adjacent frames. For the proposed model, the propagation technique introduced in Section 4.3 is used. The propagation techniques are balanced when the proposed model's propagation algorithm's spread value is $s = \frac{k}{2}$, which means that all the models need to generate the same amount of watermarks to watermark a specific number of video frames. Since the propagation algorithms can be balanced by selecting a fitting s and k, the GFLOPs for minimum number of frames that need to be watermarked is measured, which, due to the architecture of the proposed model, is 3. Therefore, during the test, all models are given 3 frames to watermark. For the baselines, 3 is the batch size and for the proposed model, 3 is the temporal dimension of the input. The results are reported in Table 7.

	HiDDeN	CIN	MBRS	Video Seal	This model
Embedder GFLOPs	67.2	49.8	96.6	126.0	47.8
Extractor GFLOPs	117.0	53.7	81.0	9.3	12.9

Table 7: GFLOPs for embedders and extractors of each model.

The proposed model demonstrates the lowest embedding cost among all tested models, with only 47.8 GFLOPs per inference pass. In comparison, the Video Seal model requires over 2.5 times more computations, reaching 126.0 GFLOPs. This highlights the architectural efficiency of the embedder for the proposed model, with the efficiency gains coming from the ability to watermark multiple frames at once. The embedder, while structurally similar to Video Seal, is not as deep, but uses 3D convolutional layers to spread the message over multiple frames as explained in Section 4.1.1.

In terms of extraction, the proposed model remains lightweight at 12.9 GFLOPs. However, it is not as minimal as Video Seal's 9.3 GFLOPs, which could be due to the use of 3D convolutional layers in the patch decoder to reconstruct the message temporally. The structure of the extractor is explained in Section 4.1.2. Furthermore, it is significantly more efficient than HiDDeN (117.0 GFLOPs) and MBRS (81.0 GFLOPs), both of which use more complex or deeper decoders. Overall, the proposed model shows great efficiency for both the embedding and extraction processes, justifying the approach of spreading watermark information across multiple frames.

5.3 Transformations Ablation

This experiment aims to determine whether the robustness performance of the model in the experiments conducted in Section 5.2.1 is primarily due to architectural design or exposure to novel transformations during training. To evaluate this, the proposed model is re-trained with these transformations excluded. The novel transformations are: AV1 Compression [DRH18], VP9 Compression [MBG+13], Sticker Overlay, Emoji Overlay, and Frame Overlay. The model is trained the same way as specified in Section 5.1 and the robustness is reported similarly to Section 5.2.1. The results for the proposed model trained with and without the novel transformations are displayed in Table 8.

	With novel transforms		Without novel transforms	
	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$
Identity	0.99	−28.9	0.99	−28.9
Valuemetric	0.98	−26.9	0.98	−26.9
Geometric	0.86	−13.2	0.86	−13.2
Overlay	0.98	−26.9	0.90	−16.7
Compression	0.92	−18.8	0.82	−10.3
Combined	0.84	−11.7	0.76	−6.8

Table 8: Robustness results for the proposed model trained with and without novel transformations.

From the results, it can be observed that the robustness for identity, valuemetric and geometric transformations is largely unchanged. This indicates that the introduction of the novel transformations during training does not affect the robustness for other transformations. For the experiments with overlay and compression transformations, there is a significant drop in performance, although the model still retains some robustness against these transformations. This suggests that the other transformations and the architecture of the model already contribute to the robustness against overlays and extra compression algorithms. For example, the geometric transformation random crop could partially contribute to the robustness against overlays, as there is a loss in watermark data when a video frame is cropped, similarly to when an emoji or a sticker is applied to a video frame. For compression, the artifacts produced by H.264 and JPEG compression can be similar to those produced by VP9 and AV1, which helps with robustness against the two novel compression algorithms. In conclusion, while the model does see a significant drop in robustness, it is still competitive with the robustness results of the baseline models as seen from Table 5. This suggests that while the novel transformations do play a significant role, other transformations and the architecture of the model also contribute to overall robustness against the novel transformations.

5.4 Real-world scenario

An experiment is conducted to evaluate the real-world effectiveness of video watermarking models. The experiment involves manually altering a set of watermarked videos, to emulate how videos could be shared on the internet. The edited videos will be uploaded to a video sharing platform,

after which they are re-downloaded and the watermark message extracted. This is motivated by the fact that video sharing platforms often use proprietary video compression algorithms, which can not be simulated in the training process. Furthermore, training time transformations can differ from real-world edits considerably.

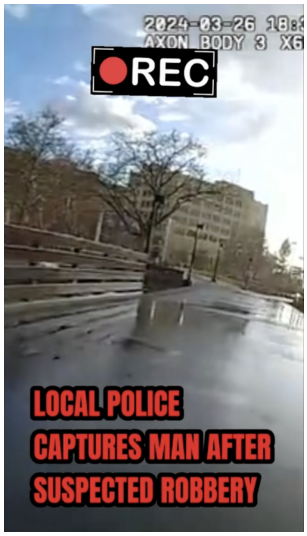
5.4.1 Set-up

YouTube was chosen as the video sharing platform to conduct the experiment on, as the platform is one of the most widely used, having more than a billion users [Kem25]. While this experiment could be conducted on multiple platforms, due to time constraints, this is an endeavour for future studies. Five user scenarios were created for the experiment, which serve as a guide for modifying the videos that are uploaded to the selected video-sharing website. These scenarios are listed in Appendix A. By describing each scenario according to the four categories of "User", "Story", "Video", and "Edits", the selection of the original video and alterations to the video are motivated. "User" identifies the kind of user sharing a video. "Story" provides context for the potential reasons why a user might share that content online. "Video" provides more details about the kind of video being shared, while "Edits" outlines the alterations made to the original video before it is uploaded.

The next stage of the experiment involves watermarking five videos selected by the user scenarios with a random binary message, which is done before any of the described alterations are made to the videos. The models evaluated in this experiment are Video Seal [FEYM24] and the novel model proposed in this thesis. Notably, the other baseline models tested in Section 5.2 are omitted, due to their poor robustness performance in the combined transformations experiment. For the proposed model, the video is watermarked according to the watermark propagation technique introduced in Section 4.3 with the spread factor $s = 2$. For Video Seal's propagation technique, the interval $k = 4$ is chosen, which was the value used in the original paper. Furthermore, this means that both models need to watermark the same number of frames as the condition $s = \frac{k}{2}$ introduced in Section 5.2.3 is satisfied[1]. The described text, pictures, and filters are superimposed on the original watermarked video using a video editor. The watermarked and edited videos are then compressed using high quality H.264 encoding (CRF = 18) in 1080p resolution at 24 frames per second. Furthermore, the videos yet to be uploaded become a baseline for the comparison. As the platform offers multiple resolutions, each resolution will be evaluated separately to compare robustness under various compression levels. An example of a video edited according to a user story is displayed in Figure 8.

[1] We have that $s = 2 \land k = 4 \land s = \frac{k}{2}$ which holds, since $2 = \frac{4}{2} \implies 2 = 2$.

(a) Unedited video, which is watermarked.



(b) Edited video according to the user story.

Figure 8: Example of a video edited according to the News Organisation user story.

5.4.2 Results

The binary message is extracted from the downloaded and baseline videos for which average bit accuracy and logarithm of the p-value are reported as defined in Section 5.2.1. The results are reported in Table 9.

	Video Seal		This model	
	Bit acc.	$\log_{10}(p)$	Bit acc.	$\log_{10}(p)$
Baseline videos	0.70	−4.56	0.65	−2.84
YouTube @ 144p	0.52	−0.42	0.49	−0.28
YouTube @ 240p	0.50	−0.28	0.53	−0.52
YouTube @ 360p	0.48	−0.21	0.52	−0.42
YouTube @ 480p	0.54	−0.62	0.50	−0.28
YouTube @ 720p	0.62	−2.04	0.57	−1.04
YouTube @ 1080p	0.68	−3.82	0.61	−1.80

Table 9: Bit accuracy of the extraction for the real-world study.

Both of the models show greatly degraded robustness when compared to the experiments in Table 5, even for the baseline video message extraction. This could be attributed to the complexity of the alterations made on the videos. For example, in Figure 8, the area cropped for the edited video is a small section of the unedited video, causing a large loss of watermark information. Furthermore, due to the aspect ratio change introduced by the cropping, when the video is resized to be passed into the extractor, a large spatial distortion is introduced between the unedited and the edited video. This indicates that while the train time transformations aid in performance, they fail to fully account for the potential real-world transformations. The train time transformations for both

models also do include temporal transformations, such as moving images or text superimposed on the video, which could also explain the degradation of robustness.

For low resolution videos (140p, 240p, 360p and 480p), both models failed to extract the original messages with the performance approximating that of random guessing. This shows the unreliability of using the models for real-world scenarios. For high resolution videos (720p and 1080p), the extraction accuracy was better, however not as good as baseline video extraction accuracy. This indicates that YouTube's proprietary compression pipeline has a significant effect on the robustness of the models, even when trained on extra video transformations such as VP9 and AV1.

While the experiment demonstrated that the models are not yet ready for real-world usage without a large amount of error correction bits, it also showed the importance of conducting similar studies in the future and the need for more temporal and spatial redundancy in the watermarks.

6 Conclusions

In this thesis, a video watermarking model was proposed to address the limitations of current models and to optimize the trade-offs between robustness, efficiency, capacity, and imperceptibility. Furthermore, novel overlay based video transformations and additional compression algorithms were introduced to the training process to refine the robustness of the proposed model. The model achieved comparable robustness results to the Video Seal model [FEYM24], while maintaining reasonable watermark imperceptibility compared to other baseline models. Though the model could benefit from a deeper architecture, the results demonstrate the effectiveness of spreading the watermark over multiple frames - including the temporal dimension in the embedding process. A reasonable efficiency gain was also demonstrated, where the proposed model exhibited the most efficient embedder and an extractor efficiency comparable to the most efficient baseline extractor, further reinforcing the potential of the temporal architecture.

An ablation study was conducted to measure the impact of the introduced transformations on the robustness of the model compared to the impact by architecture. It was found that while the novel transformations do play a significant role, other transformations and the architecture of the model also contribute to overall robustness against the novel transformations. This shows that it is beneficial to include the transformations in the training process for future models, although the architecture of the model is equally important to achieve good robustness.

A pilot study was conducted to evaluate the robustness of the video watermarking models for real-world scenarios. The results showed that while the current models are able to extract messages from heavily modified video content, they are not robust enough to be reliable for widespread use. The models especially struggled under low resolutions and heavy compressions where the accuracy approached that of a random guess.

The research questions are addressed directly:

How do overlay transformations and compression techniques impact the robustness and imperceptibility trade-off of the model and performance in real-world scenarios?

The additional video transformations and compression techniques positively affect the robustness of the proposed watermarking model, while allowing the model to maintain reasonable imperceptibility. However, since the proposed model does not include a typical GAN style discriminator, it is difficult to say if the imperceptibility loss was induced by the additional transformations or due to the training objectives. Furthermore, the proposed model did not achieve better results for the real-world experiment than the other baseline, which was trained without the additional transformations. While the results could be due to the architecture of the model, this still suggests that more complex video transformations should be included in the training for future models.

How does utilizing the temporal dimension by spreading the watermark across multiple frames impact the trade-off between robustness, imperceptibility, and efficiency in a watermarking model?

The proposed model, by utilizing the temporal dimension, achieved comparable robustness results to top performing baseline models, while significantly reducing the computational cost. While the watermarks generated by the model are not as imperceptible as the best performing baseline models, this cannot be attributed to the architecture alone, as training objectives and extra transformations also contribute to the loss in imperceptibility to not sacrifice the robustness.

How do overlay transformations, compression techniques, multi-frame embedding and model size affect the robustness, imperceptibility, and efficiency of deep learning based video steganography models?

Overall, this thesis confirms that multi-frame embedding and extended transformations mitigate some of the key limitations in deep learning based video watermarking, allowing for better balancing of the key trade-offs. However, scalability to real-world platforms remains challenging.

6.1 Further Research

Further research could focus on developing a model with variable-length messages, which could be achieved by scaling the message length based on the number of frames the message is encoded in. In the future, more in-depth hyper-parameter tuning could also be conducted to improve imperceptibility and extraction accuracy of the proposed model, as this work had hardware and time limitations.

The real-world scenario study in Section 5.4 shows the gaps in robustness for current watermarking models, particularly under heavy platform specific compression and broad video edits. Large-scale experiments replicating these conditions should be conducted for evaluating model robustness in the future. Furthermore, more complex and temporal transformations could be included in the training process, to account for complex video edits of the real world.

References

[AA24] P Aberna and Loganathan Agilandeeswari. Digital image and video watermarking: methodologies, attacks, applications, and future directions. *Multimedia Tools and Applications*, 83(2):5531–5591, 2024.

[AP17] Md Asikuzzaman and Mark R Pickering. An overview of digital video watermarking. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2131–2153, 2017.

[BAC23] Tu Bui, Shruti Agarwal, and John Collomosse. Trustmark: Universal watermarking for arbitrary resolution images. *arXiv preprint arXiv:2311.18297*, 2023.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[DD03] Gwenael Doerr and Jean-Luc Dugelay. A guide tour of video watermarking. *Signal processing: Image communication*, 18(4):263–282, 2003.

[Dee24] Google DeepMind. Veo: Video generation and editing with natural language. https://deepmind.google/models/veo/, 2024. Accessed: 2025-06-01.

[DRH18] Peter De Rivaz and Jack Haughton. Av1 bitstream & decoding process specification. *The Alliance for Open Media*, 681:1–681, 2018.

[FEYM24] Pierre Fernandez, Hady Elsahar, I Zeki Yalniz, and Alexandre Mourachko. Video seal: Open and efficient video watermarking. *arXiv preprint arXiv:2412.09492*, 2024.

[HF24] Xiangwei He and Lijuan Fang. Regulatory challenges in synthetic media governance: Policy frameworks for ai-generated content across image, video, and social platforms. *Journal of Robotic Process Automation, AI Integration, and Workflow Optimization*, 9(12):36–54, 2024.

[HWZ+22] Mingze He, Hongxia Wang, Fei Zhang, Sani M Abdullahi, and Ling Yang. Robust blind video watermarking against geometric deformations and online video sharing platform processing. *IEEE Transactions on Dependable and Secure Computing*, 20(6):4702–4718, 2022.

[HZL+25] Runyi Hu, Jie Zhang, Yiming Li, Jiwei Li, Qing Guo, Han Qiu, and Tianwei Zhang. Videoshield: Regulating diffusion-based video generation models via watermarking. *arXiv preprint arXiv:2501.14195*, 2025.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JFZ21] Zhaoyang Jia, Han Fang, and Weiming Zhang. Mbrs: Enhancing robustness of dnn-based watermarking by mini-batch of real and simulated jpeg compression. In *Proceedings of the 29th ACM international conference on multimedia*, pages 41–49, 2021.

[Kem25] Simon Kemp. Digital 2024 april global statshot report - datareportal – global digital insights, Mar 2025.

[KMP+22] Eelandula Kumaraswamy, Kommabatla Mahender, Ch Rajendra Prasad, N Govardhan, and Bonthala Prabhanjan Yadav. Digital watermarking techniques: Comparative analysis and robustness for real time applications. In *AIP Conference Proceedings*, volume 2418. AIP Publishing, 2022.

[KP22] Maciej Kaczyński and Zbigniew Piotrowski. High-quality video watermarking based on deep neural networks and adjustable subsquares properties algorithm. *Sensors*, 22(14):5376, 2022.

[KPP22] Maciej Kaczyński, Zbigniew Piotrowski, and Dymitr Pietrow. High-quality video watermarking based on deep neural networks for video with hevc compression. *Sensors*, 22(19):7552, 2022.

[LBBH02] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[LH17] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[LLC+23] Xiyang Luo, Yinxiao Li, Huiwen Chang, Ce Liu, Peyman Milanfar, and Feng Yang. Dvmark: a deep multiscale framework for video watermarking. *IEEE Transactions on Image Processing*, 2023.

[LMGH22] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, pages 280–296. Springer, 2022.

[LRD21] S Bhargavi Latha, D Venkata Reddy, and A Damodaram. Video watermarking using neural networks. *International Journal of Information and Computer Security*, 14(1):40–59, 2021.

[LWW+24] Lina Lin, Deyang Wu, Jiayan Wang, Yanli Chen, Xinpeng Zhang, and Hanzhou Wu. Automatic, robust and blind video watermarking resisting camera recording. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.

[LZL+24] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models. *arXiv preprint arXiv:2402.17177*, 2024.

[MBG+13] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. The latest open-source video codec vp9-an overview and preliminary results. In *2013 Picture Coding Symposium (PCS)*, pages 390–393. IEEE, 2013.

[MG18] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[MGH+22] Rui Ma, Mengxi Guo, Yi Hou, Fan Yang, Yuan Li, Huizhu Jia, and Xiaodong Xie. Towards blind watermarking: Combining invertible and non-invertible mechanisms. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1532–1542, 2022.

[MKNI19] Aayush Mishra, Suraj Kumar, Aditya Nigam, and Saiful Islam. Vstegnet: Video steganography network using spatio-temporal features and micro-bottleneck. In *BMVC*, volume 274, 2019.

[MKR+22] David Megías, Minoru Kuribayashi, Andrea Rosales, Krzysztof Cabaj, and Wojciech Mazurczyk. Architecture of a fake news detection system combining digital watermarking, signal processing, and machine learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 2022, 13 (1): 33-55,*, 2022.

[Ope24] OpenMoji. Openmoji: Open source emojis for designers, developers and everyone else!, 2024. Accessed: 2025-05-11.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[RGH+24] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[Ric11] Iain E Richardson. *The H. 264 advanced video compression standard*. John Wiley & Sons, 2011.

[RvDK25] Bram Rijsbosch, Gijs van Dijck, and Konrad Kollnig. Adoption of watermarking for generative ai systems in practice and implications under the new eu ai act. *arXiv preprint arXiv:2503.18156*, 2025.

[SB24] Anuradha Saini and Sushil Bhardwaj. A review on digital video watermarking security: Significance and persistent challenges. In *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, pages 1–8. IEEE, 2024.

[SMC21] S Shyam Sundar, Maria D Molina, and Eugene Cho. Seeing is believing: Is video modality more powerful in spreading fake news via online messaging apps? *Journal of Computer-Mediated Communication*, 26(6):301–319, 2021.

[VNS24] Fatimetou Abdou Vadhil, Mohamedade Farouk Nanne, and Mohamed Lemine Salihi. The powerful ai: An exploration of generative artificial intelligence taxonomy and applications. In *International Conference on Artificial Intelligence and its Applications in the Age of Digital Transformation*, pages 236–250. Springer, 2024.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[WTBR21] Chloe Wittenberg, Ben M Tappin, Adam J Berinsky, and David G Rand. The (minimal) persuasive advantage of political video over text. *Proceedings of the National Academy of Sciences*, 118(47):e2114388118, 2021.

[WWW23] Baowei Wang, Yufeng Wu, and Guiling Wang. Adaptor: Improving the robustness and imperceptibility of watermarking by the adaptive strength factor. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(11):6260–6272, 2023.

[YGW+23] Guanhui Ye, Jiashi Gao, Yuchen Wang, Liyan Song, and Xuetao Wei. Itov: efficiently adapting deep learning-based image watermarking to video watermarking. In *2023 International Conference on Culture-Oriented Science and Technology (CoST)*, pages 192–197. IEEE, 2023.

[YWZ+24] Ling Yang, Hongxia Wang, Yulin Zhang, Mingze He, and Jinhe Li. An adaptive video watermarking robust to social platform transcoding and hybrid attacks. *Signal Processing*, 224:109588, 2024.

[ZGC+24] Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, et al. Sok: Watermarking for ai-generated content. *arXiv preprint arXiv:2411.18479*, 2024.

[Zha22] Tao Zhang. Deepfake generation and detection, a survey. *Multimedia Tools and Applications*, 81(5):6259–6276, 2022.

[ZHQM07] Jing Zhang, Anthony TS Ho, Gang Qiu, and Pina Marziliano. Robust video watermarking of h. 264/avc. *IEEE transactions on circuits and systems II: express briefs*, 54(2):205–209, 2007.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[ZKJFF18] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018.

[ZNSL23] Yulin Zhang, Jiangqun Ni, Wenkang Su, and Xin Liao. A novel deep video watermarking framework with enhanced robustness to h. 264/avc compression. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 8095–8104, 2023.

[ZWH+24] Fei Zhang, Hongxia Wang, Mingze He, Ling Yang, and Jinhe Li. Adaptive video watermarking with perceptual guarantee and efficiency optimization. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4610–4614. IEEE, 2024.

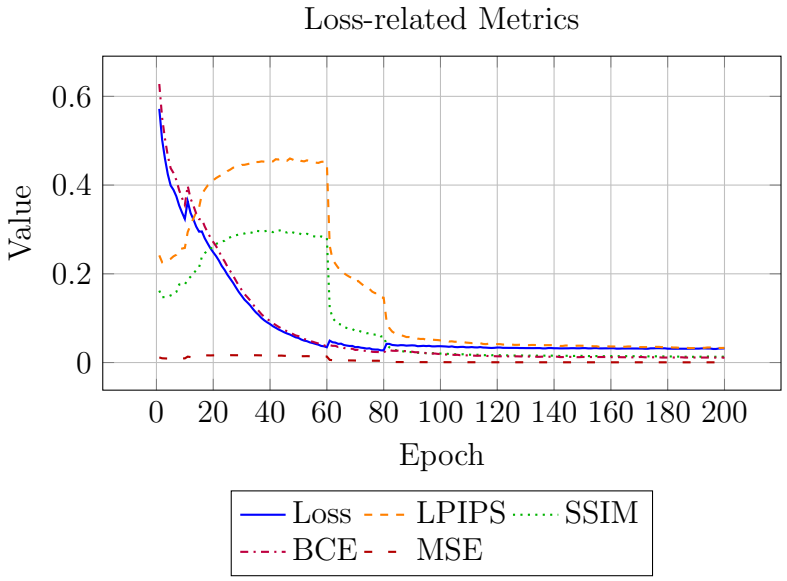
[ZXCIV19] Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019.

A User Stories Table

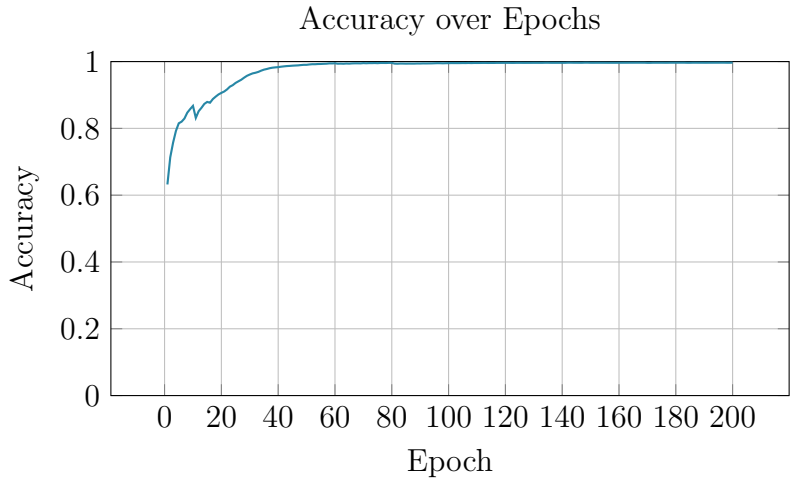
User	Story	Video	Edits
News Organisation	The user discovers police footage showing the arrest of a suspected criminal. To inform their audience, they aim to create a news segment using this video	A low-resolution video captured from a body camera showing an arrest in progress	In order to enhance viewer engagement, the user overlays a red circle around the individual committing the robbery. In addition, they superimpose a short and captivating caption to attract interest and convey the message of the video
Political party	Seeking to influence public opinion for an upcoming election, the user searches for video material that portrays a political opponent unfavourably	An AI-generated video filmed from a concealed smartphone perspective, depicting an opposition politician accepting a bribe	To clarify the events that transpire in the video, the user adds an animated arrow pointing at the politician when they take the bribe. The logo of the political organisation is also added as well as some on-screen text summarizing the events and urging viewers to support their political party
Animal Rights Organisation	The user seeks to raise awareness about the poor conditions that captive animals live in	They find a video depicting the harsh treatment of an orca that is held captive in an animal park. The video is filmed from someone in the audience of a live show, where a staff member is seen to mistreat the orca	The original video includes a warm color filter. The user wants to enhance clarity by adding a magnified circle to highlight the mistreatment scene. There is also a faint logo of the organisation in one corner, along with some text at the top, describing the incident
An elderly individual	This user frequently browses social media to find entertaining videos to share with their grandchildren or all their followers	The user finds an AI-generated video of a puppy playing with a ball in a park and decide to repost it	They add a color filter to the video, as well as some text. Due to the format discrepancies between the original video and the social media platform, the video only takes up about two-thirds of the screen
A video game enthusiast	This person is an avid fan of video games, and therefore regularly uses social media to consume video game-related content. In anticipation of a new video game release, they search for related footage	They find a YouTube video in landscape orientation of someone playing an early version of the game and decide to repost it on their social media	As the video is originally in landscape, the person crops the video to fit the vertical format of the social media platform. They also add a visual filter, insert enthusiastic text and add stickers to express their excitement for the release of the game

Table 10: Overview of user stories and associated video edits for the real-world experiment.

B Proposed Model Training Plots



(a) Loss, LPIPS, SSIM, BCE, MSE



(b) Accuracy

Figure 9: Training Metrics Over Epochs