BEYOND THE PIXELS: LEARNING AND UTILISING VIDEO COMPRESSION FEATURES FOR LOCALISATION OF DIGITAL TAMPERING

PAMELA JOHNSTON

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE
SCHOOL OF COMPUTING SCIENCE AND DIGITAL MEDIA
ROBERT GORDON UNIVERSITY
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

August 2019

Supervisor Dr. Eyad Elyan

Abstract

Video compression is pervasive in digital society. With rising usage of deep convolutional neural networks (CNNs) in the fields of computer vision, video analysis and video tampering detection, it is important to investigate how patterns invisible to human eyes may be influencing modern computer vision techniques and how they can be used advantageously.

This work thoroughly explores how video compression influences accuracy of CNNs, and shows how optimal performance is achieved when compression levels in the training set closely match those of the test set. A novel method is then developed, using CNNs, to derive compression features directly from the pixels of video frames. It is then shown that these features can be readily used to detect inauthentic video content with good accuracy across multiple different video tampering techniques. Moreover, the explainability of these features allows predictions to be made about their effectiveness against future tampering methods.

The problem is motivated with a novel investigation into recent video manipulation methods, which shows that there is a consistent drive to produce convincing, photo-realistic, manipulated or synthetic video. Humans, blind to the presence of video tampering, are also blind to the *type* of tampering. New detection techniques are required, and in order to compensate for human limitations, they should be broadly applicable to multiple tampering types. This thesis details the steps necessary to develop and evaluate such techniques.

Keywords: Computer vision; video compression; deep learning; tampering detection; video tampering.

Acknowledgements

Special thanks must go to my principal supervisor Dr. Eyad Elyan for his inspiration, guidance in all things academic, and his kind reassurances and motivation, given freely when required. Thanks also go to the rest of my supervisory team, particularly Prof. Chrisina Jayne for her mentorship and insight. And to the staff of the School of Computing at RGU who were always there to help and advise.

I would also like to express my grateful appreciation of all my research buddies, past and present. Ahmed and Jérémie for lighting the path just ahead. Kyle and Adamu for accompanying me on the journey, Scott for sensible discussions on coding, and Pat for showing me that imbalanced data really does pose a big problem.

I am also exceptionally grateful to my family for their endless love and support. Thanks, in particular, go to my husband Barry for his support and patience with me in this research; to my boys Hamish and Fergus for their enthusiasm; and to my mum for providing both parental pride and child care. This would not have been possible without you.

Abbreviations

AVC	Advanced Video Codec (video format)
BPG	Better Portable Graphics (image format)
bpp	bits per pixel
CBR	Constant Bitrate
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
FFW	“Fake Faces in the Wild”
FN	False Negative
FP	False Positive
fps	Frames Per Second
GAN	Generative Adversarial Network
GOP	Group of Pictures
HD	High Definition
JPEG	Joint Pictures Expert Group (image format)
LSTM	Long Short Term Memory
mAP	mean Average Precision
NA	Network Architecture
PRNU	Photo Response Non Uniformity
PSNR	Peak Signal to Noise Ration
QF	Quality Factor
QP	Quantisation Parameter
RF	Random Forest
RNN	Recurrent Neural Network
SD	Standard Definition
SSIM	Structural SIMilarity
SVM	Support Vector Machine
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
VBR	Variable Bitrate

Summary of Publications

The author published four articles as part of the work in this thesis. These were:

Journals

- Pamela Johnston and Eyad Elyan. "A review of digital video tampering: From simple editing to full synthesis", *Digital Investigation*, Elsevier, 2019. https://doi.org/10.1016/j.diin.2019.03.006
- Pamela Johnston, Eyad Elyan, and Chrisina Jayne. "Video tampering localisation using features learned from authentic content", *Neural Computing and Applications*, Springer, 2019. https://doi.org/10.1007/s00521-019-04272-z

Conferences

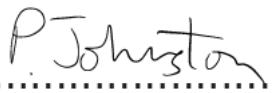
- Pamela Johnston, Eyad Elyan, and Chrisina Jayne. "Spatial Effects of Video Compression on Classification in Convolutional Neural Network" in *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1370 - 1377. IEEE, 2018. https://doi.org/10.1109/IJCNN.2018.8489370
- Pamela Johnston, Eyad Elyan, and Chrisina Jayne. "Toward video tampering exposure: Inferring compression parameters from pixels" in *19th International Conference on Engineering Applications of Neural Networks (EANN)*, pages 44 - 57. Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-98204-5_4

Code

- The author's github is available at https://github.com/pamelaajohnston

Declaration

I confirm that the work contained in this PhD project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed:  Date: August 2019

Pamela Johnston

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the world of video analysis, video compression is often over looked. So much of the online content we view is compressed that compression itself is seen as inherent. This is almost as true for machine eyes as it is for human eyes. Many deep learning techniques rely on vast quantities of data, harvested from publicly available online sources and, as such, subject to compression. Some techniques, particularly in the area of video forensics, list compression or reduced bit-rate video as one of the open challenges. Video tampering detectors, in particular, often show reduced performance when faced with video that has been compressed or recompressed post-tampering. This work aims to look *beyond the pixels* and examine the features and effects produced by video compression, how they can be detected and whether they can be usefully applied to help address some of the problems attributed to them. We aim to ascertain whether compression features learned from authentic and controlled video sources can ultimately aid the detection of inauthentic or manipulated video.

1.1 Compression, Analysis and Manipulation

Video is a medium which is particularly rich in data, with more hours of video publicly available online than any single person could hope to watch in their lifetime. To overcome the problem of storing and distributing so much content, online video is almost always compressed using some standardised algorithm. Moreover, many videos are compressed and recompressed as they are moved between platforms and locations online. The end result is that viewers often see a simulacrum of the video that was originally captured on the camera sensor. Modern video compression algorithms have been designed and revised over decades so that they can efficiently compress large

quantities of pixel data. To do this, they exploit weaknesses in the human visual system. Video sequences are never left completely unchanged by lossy compression, but changes may simply fall beneath the threshold of human perception. What is currently unclear, is how much compression, and thus the exploitation of human vision weakness, affects the development of machine vision. Compression affects the vast majority of videos available online, therefore any machine learning techniques which utilise this enormous mine of data will find themselves also subject to the effects of compression.

Video analysis is a broad field. It covers many subjects such as: video forensics [1, 2], object detection [3], visual object tracking [4] and video classification [5]. More recently, video manipulation [6, 7], frame interpolation and prediction [8, 9, 10] and super resolution [11] have become popular. Just as in image analysis and processing, recent trends in video have brought deep learning to the fore. In the past, features for video and image analysis were hand-crafted and based on mathematical understanding of what might prove useful or overcome current challenges. The Scale Invariant Feature Transform or SIFT [12], for example, was designed to overcome the way that images and the objects within them can be rescaled to different sizes. More recently, deep learning has risen through the ranks in both image and video analysis. With the seminal success of AlexNet [13] on the ImageNet dataset [14] in 2012 and the rise of powerful parallel computing processors in the form of GPUs, the focus of video and image analysis has moved to deep learning features that emerge from large quantities of data. The challenge of variations in object sizes, for example, is now met simply by including many examples of different objects at different scales in the training dataset. While they have proven very effective, deep learning methods are data hungry. This inhibits their application to fields where there are few or limited examples.

The large quantity of available video and video analysis techniques also evidences the importance that society places on video content. This serves to demonstrate the vast potential that the video medium has for societal influence. A single sequence on YouTube can be viewed millions of times, and its authenticity may be taken for granted. As video manipulation techniques advance, it is vital to develop machine vision tools which can compensate for human blindness and are capable of gauging the authenticity or forensically analysing a given sequence.

In the past, video tampering was limited to inter-frame forgery and intra-frame or object forgery. Inter-frame forgery is where scenes of a video are carefully joined together to create a false order of authentic frames. Intra-frame forgery usually involves splicing content from two sequences together or inpainting to conceal a particular object. Each individual *frame* in an intra-frame tampered sequence may contain authentic and

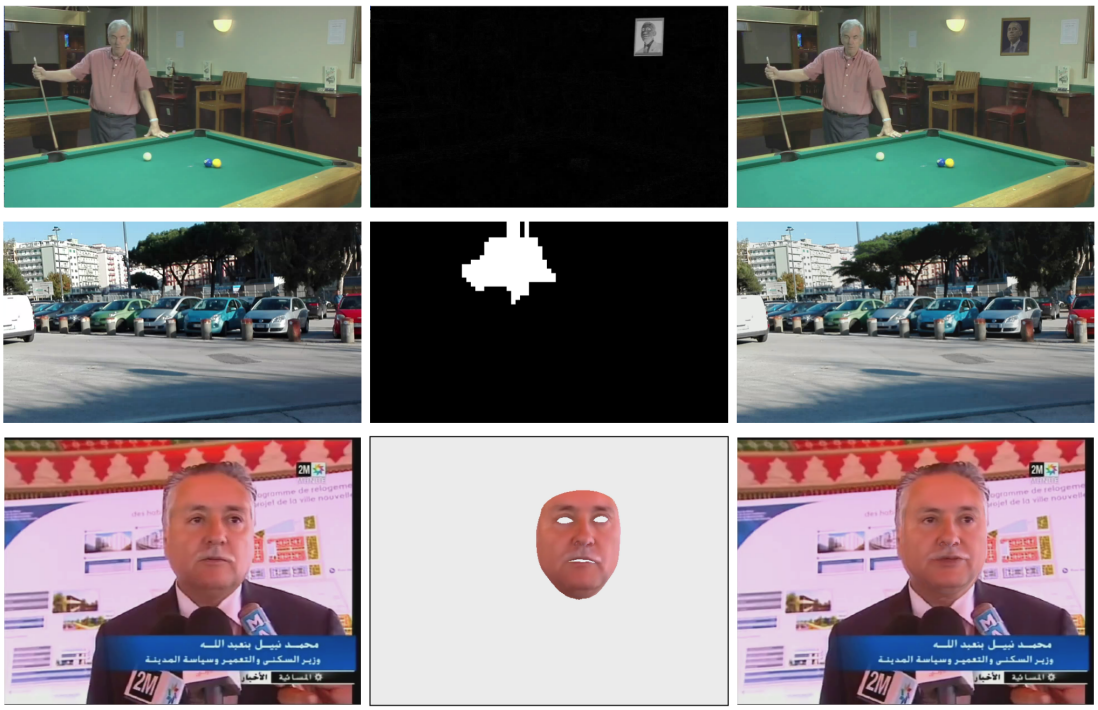


Figure 1.1: Examples of tampered and authentic frames and their masks from datasets [15, 16, 17]. The top sequence from [15] consists of an added picture on the wall and is an example of splicing. The middle sequence from [16] has had a tree added to it and is an example of a spatio-temporal copy-move. The bottom sequence from [17] is an example of digital re-enactment or digital puppetry, and subtle changes to the facial expression have been made, which are most obvious around the shape of the mouth and eyes. The technique used in [17] was found to be almost undetectable to humans.

tampered pixels. Video manipulation, however, is a rapidly expanding field, with many new techniques capable of forging realistic video content (See Chapter 3). Figure 1.1 illustrates how visually convincing some of these methods are by showing some examples of tampered frames from available datasets. Digital re-enactment or digital puppetry involves changing the facial expressions of a person in a pre-recorded video to follow the actions of an actor in a different video. This allows content producers to create forged videos from authentic content. Recent discoveries in video forgery have show that some manipulation techniques leave humans completely unable to differentiate between authentic content and digital re-enactment [17]. Deep neural networks trained to differentiate between content processed in a particular way and unprocessed content, however, are almost unreasonably accurate [17, 2, 18]. This implies that these forms of processing leave some sort of "fingerprint" on the pixels themselves, and that although this fingerprint is invisible to humans, DNNs have no trouble uncovering it. The question is whether any facet of the processing fingerprints can be generalised to apply to different datasets.

1.2 Motivation

The main question that this thesis serves to answer is: can video compression features be used as an ally rather than an enemy? Video compression is explicitly reported as a challenge in many fields. The authors of [19] showed how JPEG compression has a negative effect on classification accuracy for images classified by deep neural networks. This effect may even have been understated given that the neural network itself was not trained with completely uncompressed data but, rather, with ImageNet, which is data harvested from online public sources. Rössler et al [17] demonstrated how compression could effectively "launder" digital re-enactment, making it less detectable for machine learning techniques trained on high quality examples. The VISION dataset [20] showed how sensor pattern noise, often used as a digital fingerprint to identify the source of a video, was more challenging to detect once the video content had been processed via uploading and downloading to common media sharing sites. With that in mind, it is worthwhile to quantify how much video compression currently affects performance in deep neural networks, and ascertain whether that knowledge can be subsequently used to improve other techniques.

Recompression is also listed as an important aspect of video forensics [21]. Although recompression does not explicitly imply the presence of video tampering, it can diminish the performance of many tampering detectors. Recompression also invalidates data that can be gathered directly from compressed bitstream syntax elements. A low quality video may be recompressed at a high bitrate but it will still retain all the visible pixel artifacts associated with the original low bitrate. Therefore, in order to utilise any knowledge gained about how compression affects performance in deep learning techniques, some method of quantifying the compression features is required.

Given the prevalence of compression features in the wild, an analysis of video compression features within the context of deep learning is a valuable addition to many research fields. If compression features can be used as an intermediate representation of video, then this may help to overcome the large data requirements of machine learning techniques when used in fields such as video forgery detection. In the field of video tampering, there are new and emerging techniques which are currently invisible to humans [17, 22]. While deep learning methods may be appropriate for tackling this challenge, they are data hungry. A large scale and varied dataset comprising numerous different tampering techniques does not yet exist. Therefore, to keep pace with tampering methods, it is more appropriate to rephrase tampering detection as authenticity validation. Photo Response Non Uniformity (PRNU) is a form of sensor pattern noise and can be used to detect some forms of tampering, particularly the splicing of

content from two different camera sources [23]. Unfortunately many techniques based on PRNU are subject to diminishing performance in the presence of video compression. If compression features can be derived directly from pixel data, it may be possible to turn the challenge of compression into an opportunity. Compression features may constitute a new form of digital forensic fingerprint, the details of which may differentiate between content from different sources, whether those sources are cameras or pixels generated by some tampering technique. If this is the case, then the first step is to derive compression features directly from pixels themselves.

1.3 Research Objectives

This thesis has the following objectives:

- Review the current trends in video manipulation and identify how these have changed in recent years with a view to evaluating the potential efficacy of features based on compression. This includes identification of publicly available datasets, how widely these are used and the challenges associated with them. This is then used to establish a set of simple recommendations to encourage future researchers in video manipulation to provide examples of their work to contribute to the development of generic tampering detection methods.
- A thorough evaluation of how video compression affects learning in deep convolutional neural networks. This will help to gain an understanding of how the challenges associated with compressed data arise.
- Create a novel method to derive compression features directly from pixel data. Videos can be recompressed and re-recompressed thus invalidating any compression features that can be extracted directly from the compressed bitstream. One way to objectively analyse a video in terms of its compression is to derive measures of compression directly from the pixels themselves. A novel method to do this using convolutional neural networks is presented in this thesis and a full evaluation of such is provided.
- To establish whether video compression features can be used to localise tampering within video content using existing, publicly available datasets. The hypothesis here is that authentic and tampered content may exhibit different compression feature distributions.

1.4 Contributions

This thesis makes the following contributions:

- An extensive investigation to identify current state-of-the-art techniques in image and video tampering and study the current provision of publicly available datasets. Although the field of video tampering *detection* has garnered a collection of extensive reviews, the field of realistic video manipulation is still a relatively new field which is worthy of study.
- An experimental framework designed to provide a full analysis of the spatial effects of video compression on classification accuracy in CNNs [24]. A number of results throughout the literature suggest that an understanding of compression may benefit CNNs for video analysis. By examining this in isolation, we provide objective conformation that knowledge of underlying compression may be beneficial for classifiers.
- A novel deep learning method of deriving compression features directly from the pixels of compressed bitstreams [25]. This involves the necessary framework for synthesis of appropriate datasets. A full evaluation is also provided, demonstrating which aspects of video compression itself inhibit the ability to accurately derive compression parameters from pixel data alone.
- Creation and evaluation of new methods for video tampering localisation using features of compression [26]. This includes identification of key frames in a sequence.

The resulting publications are listed in the preface section “Publications”.

1.5 Thesis Structure

Chapter 2 provides an introduction to the relevant research fields used in this thesis, covering both deep learning and the basics of standard video compression. Chapter 3 provides the motivation for research into compression-based features by detailing current trends in video tampering and revealing the need for technique agnostic video tampering detection: a need that can be fulfilled through the use of compression features.

Chapter 4 details how video compression affects classification in CNNs, and provides evidence for the usefulness of compression features estimated directly from the pixels.

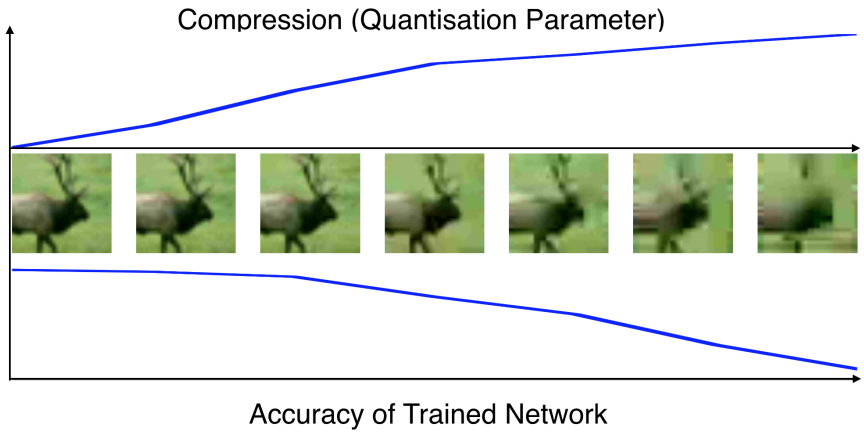


Figure 1.2: Summary of how higher levels of compression in training sets affects accuracy of trained CNNs

Figure 1.2 summarises the intuitive findings that, in general, CNNs trained on more highly compressed data have poorer accuracy. However CNNs also perform optimally when the training and test data share similar levels of compression. It is therefore advantageous to somehow measure levels of compression.

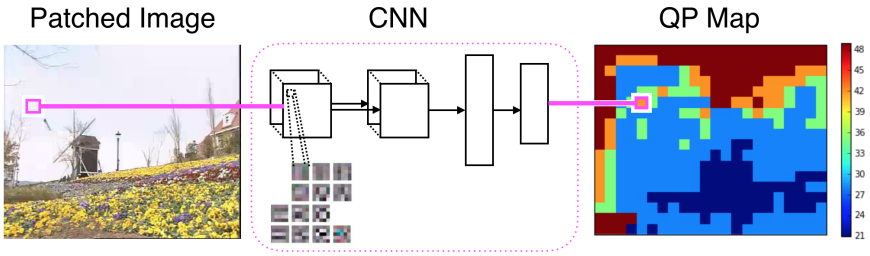


Figure 1.3: CNNs can be used to learn quantisation parameters from pixels, thus translating frame data to a compression fingerprint

Chapter 5 presents a novel method to extract compression features, such as quantisation, directly from pixels using CNNs. Figure 1.3 summarises one of these methods. Chapter 6 shows how this can be extended to other compression features and how these can be used to identify key frames in a video sequence.

Chapter 7 examines how compression features predicted from the pixels can be utilised to aid tampering localisation in publicly available tampered datasets. Figure 1.4 gives examples of this.

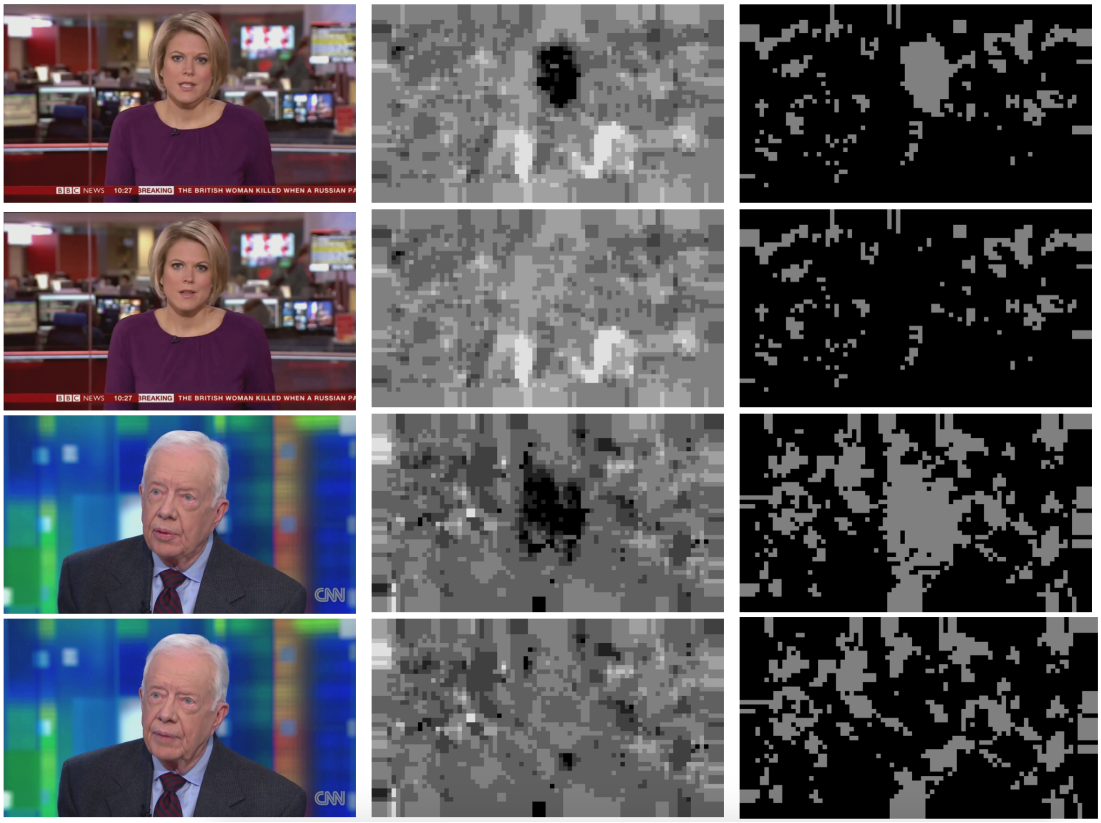


Figure 1.4: Using features of compression to represent video frames can aid in tampering localisation. Frames are from [17]. The first column shows still images from related sequences. The top example of each pair is tampered using digital puppetry on the face region only, the bottom example of each pair is authentic. Tampered and authentic frames are almost indistinguishable to the naked eye. The middle column shows quantisation as estimated by a trained CNN; darker areas are considered less compressed. The tampered content content registers as uncompressed. The third column shows the results of a classifier trained to differentiate between authentic and tampered content. Full details of this complete process are given in subsequent chapters.

Chapter 2

Research Background

This chapter provides a necessary grounding in video compression and deep neural networks. Video compression is a multi-variable process and this chapter illustrates and explains some of the effects of the compression process that may go unnoticed to the untrained eye. Video compression standards are prescriptive, but, as shown in this chapter, there are sufficient degrees of freedom that may allow individual encoders to leave their own fingerprint upon the pixels that pass through them. Deep neural networks are a useful tool in the field of computer vision and particularly well suited to analysing video frames. In order to make optimal use of them, it is important to have a good understanding of the mechanisms that drive them.

2.1 Standard Video Compression

The majority of video available online is compressed. Compression allows more content to fit into the same space. There are many ways to compress a video file, but far fewer industrial standards. The most commonly used video codecs form part of the MPEG series of standards. In this section, we cover the basics of video compression that apply to the industrial standards H.264/AVC [27] and H.262/MPEG-2 part 2 [28].

Table 2.1 gives the names of relevant industry standards. MPEG-4 is a series of standards which define how audio and video data should be compressed and transmitted. It encompasses a number of distinct methods of video compression including: MPEG-4 Part 2, which is related to H.263 [29]; and MPEG-4 Part 10 which is also known as H.264 or Advanced Video Codec (AVC). MPEG-4 also governs how compressed video should be packaged as a file and packetised for streaming purposes. MPEG-2

Table 2.1: Industry standards related to video

Abbreviation	Other names	Definition
MPEG-2 part 2	H.262 [28]	A standard governing video compression
MPEG-4	-	A series of documents governing digital video and audio compression
MPEG-4 part 10	H.264/AVC [27]	A standard governing video compression
MPEG-4 part 2	related to H.263 [29]	A standard governing the video compression

part 2 is a precursor for H.264/AVC and they use many similar mechanisms.

Video compression relies on reducing the level of redundancy within a sequence of pixels. In essence, it involves a pre-defined method of making predictions from available data and a series of syntax elements which correct these predictions. The first frame of a sequence will always be encoded from without any reference data. This is called an **intra**, **key** or I- frame (Table 2.2 defines some useful compression terms). An intra frame relies only on data within itself to be completely decoded. Subsequent frames can be predicted from the intra frame or from each other and these are called **inter**, **predicted** or P- or B- frames. Inter frames rely on reference pictures. Optimally predicting every new pixel from past pixels is the most efficient form of compression and will yield the smallest number of bits, however it is also important to provide access points into the bitstream. These can be provided by inserting key frames periodically throughout the sequence. Access points can also be provided by distributing intra-only regions over a number of frames and instructing the decoder to display only when the entire frame is assembled (as available in [30]). Key frames define what is known as a Group of Pictures (GOP). A GOP starts with a key frame and ends on the last predicted frame before the next access point. GOPs are self contained. In H.264/AVC, it is possible to have intra frames that are not key frames because reference pictures may be kept beyond these intra pictures, and the interested reader may refer to [31] for a complete overview of the H.264/AVC standard. For simplicity, here we consider every intra frame to be the start of a GOP. Figure 2.1 shows a GOP construction. The GOP starts with an I-frame which can be decoded alone. The next frame to be encoded is the nearest P-frame, which references the data in the I-frame. Once the I- and P-frames are encoded, the data within them can be used to encode the B-frames. Note that the encoding order is not necessarily the same as the display order. Of the three different frame types, B-frames are the most efficiently compressed but they require reordering of frames prior to encoding, and consequent reordering prior to display on

Table 2.2: Video compression terms

Term	Definition
GOP	Group of Pictures. A whole number of frames starting with an I-frame.
Intra frame	A key frame which can be decoded by itself.
Inter frame	A predicted frame which requires data from other frame(s) before it in the bitstream.
Reference frame	A decoded frame used as reference for a predicted frame
I-	Intra.
P-	Inter, single direction (only uses frames temporally before it).
B-	Inter, bi-directional (uses frames temporally before and after)
Macroblock	A group of pixels, 16x16 pixels in the Y channel plus corresponding U and V channels
Slice	A group of macroblocks
DCT	Discrete Cosine Transform, for spatial to frequency domain transform
residual	The difference between a predicted and an actual value
Motion vectors	An x- and y- vector which define a region in a reference frame to use as a prediction
fps	Frames Per Second
field	half a frame, alternate lines horizontally, used in interlaced sequences
progressive	Consisting of compelete frames. The “p” in 720p, 1080p
interlaced	Consisting of fields. The “i” in 1080i
CBR	Constant bitrate, a common rate control method
VBR	Variable bitrate, a rate control method which simplifies encoder operation

the decoder side.

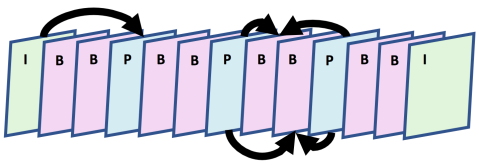


Figure 2.1: An example Group of Pictures (GOP) structure

GOP structure is decided by individual encoders and can vary wildly depending on requirements. In MPEG-2 and other standards prior to H.264, the standard itself defined a maximum period between intra frames. This was due to non-integer based arithmatic used within the frequency transforms. Rounding errors introduced by this could cause drift between encoder and decoder. This drift was held in check by a defined maximum number of frames between intra frames. In H.264/AVC, the frequency transforms were integer based and, as such, completely reversible. This meant that

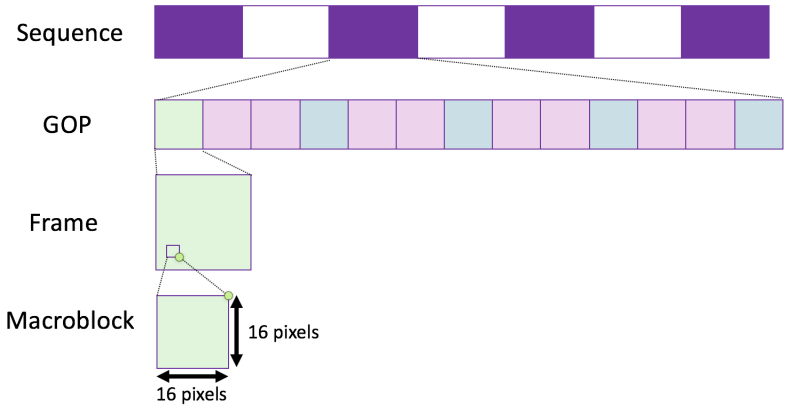


Figure 2.2: Compression heirarchy: from sequence to pixels

in H.264/AVC, GOP structure essentially became a balance between compression efficiency and stream integrity. Encoding every frame as an intra frame, for example, means that every frame in the sequence can be decoded individually and lost data has minimal visible impact, but the bitrate will be necessarily high. Encoding an entire sequence as a single GOP with only one intra frame at the beginning will improve the compression efficiency and lower the bitrate substantially, but if any part of the sequence is lost or corrupted, then there is no way to recover from and all frames from the lost data onwards must be discarded. GOP structure can also rely on content. Intra frames, for example, are the best means of encoding the first frame after a cut scene, but this relies on the encoder's ability to buffer and analyse the content of a sequence. In some encoder implementations, the GOP structure is fixed and does not depend on sequence content at all.

Figure 2.2 shows the main objects used in compression codecs H.264/AVC and MPEG-2. A sequence is formed from GOPs; GOPs are made up of individual frames; and frames consist of macroblocks, which are 16x16 pixel squares. For stream packetisation purposes, macroblocks can also be grouped into slices and in certain modes, slices can be decoded completely independently of each other thus making the bitstream more robust against packet loss. Although macroblocks can be further divided into sub-macroblocks in H.264/AVC, for simplicity we will consider the macroblock as the base unit of compression. For more information on these modes, the interested reader is referred to [31].

Figure 2.3 shows the basic processes in a standard encoder. The incoming frame is first divided into macroblocks which are typically processed in raster scan order. In intra frames, the first macroblock is stand alone as there is not yet any data to use as a reference. In H.264/AVC, subsequent macroblocks in an intra frame can

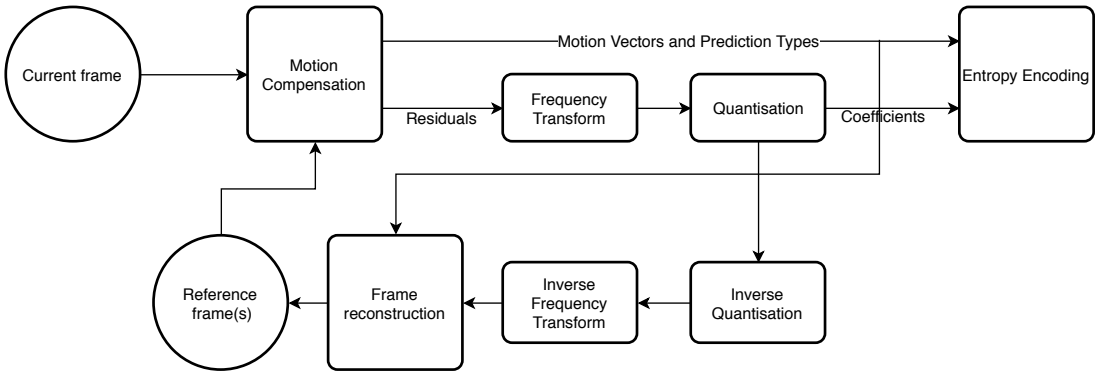


Figure 2.3: Encoder flowchart

then be encoded using data from immediately adjacent neighbours to the left and from the row above if they are available. A prediction is made from the available neighbours. Different macroblock *types* are used to define a specific combination of neighbouring data to make a prediction. For example, a prediction might involve using pixels from the left neighbour only, or it might use macroblocks from the row above. The *difference* between this prediction and the actual macroblock is taken as the residual. This is an efficient process because often macroblocks share their appearance with their neighbours. The residual is then frequency transformed using Discrete Cosine Transform (DCT), quantised and the resulting coefficients and syntax elements entropy encoded ready for transmission. Meanwhile, the reverse path of the encoder takes the quantised coefficients and dequantises them, transforms them back into the spatial domain and reconstructs the macroblock pixels. Reconstructed macroblocks form the reconstructed frame, which can then be used as a reference frame. H.264/AVC maintains lists of reference frames, although the standard defines how these changes are communicated to decoders via the bitstream, reference list selection and management is left up to individual encoder implementations..

To encode an inter frame, the first step is to locate a good match from the data in the available reference frames. It should be noted that the reference frames are the *reconstructed* frames, not the original uncompressed frames. A good match is found through motion estimation and various methods of block based motion estimation have been suggested in the literature [32]. In MPEG-2, motion estimation was accurate to half pixels with standard-defined means of interpolating between pixels [28]. In H.264/AVC, it is accurate to quarter pixels [27], allowing more compression efficiency at the expense of processing overheads. Motion estimation mechanisms themselves are not part of the industrial standards, and it is up to individual encoders to implement a method that finds an appropriate match. One method is to individually compare the current macroblock to different regions in the reference frames and minimise the sum

of absolute differences. Methods of reference region selection have been examined in detail [32, 33, 34] because as compression standards advance, exhaustive search becomes impossible. If an appropriate match is located, then the residual is calculated and encoding proceeds as in the intra case. Alternatively, the macroblock can be encoded as an intra block, in the same way as for the macroblocks in an intra frame. In some cases, if there is no residual and the macroblock matches the prediction well enough, the macroblock can be *skipped*, and this is the smallest type of macroblock. A run of skipped macroblocks can be encoded as a single syntax elements, thereby allowing individual skipped macroblock to occupy less than a single bit in the bitstream.

In inter frames, consideration must be given to minimise not only the number of bits used to represent the residual, but also the number of bits given over to motion vectors. Motion vectors themselves are predicted using the motion vectors of available neighbouring macroblocks, and the difference between the predicted motion vectors and the actual motion vectors added to the bitstream. Sometimes the most favourable match is not the one that generates the smallest residual, but one where the combination of encoded motion vector differences and residual generates the fewest number of bits. This is known as rate distortion optimisation [34]. With so many options available it is unsurprising that individual encoder implementations produce slightly different bitstreams.

2.1.1 Frequency Domain

Frequency domain transform is commonly used in video and image compression. The human eye is more sensitive to noise and artifacts in areas of low frequency [35]. A person might be able to spot a single black bird in a uniformly blue sky, but they are unlikely to spot a small reduction in the number of hairs on the head of a dog. Therefore, the lossy part of compression takes place in the frequency domain by quantising frequency coefficients.

For non-predicted data, the pixel data itself is transformed into the frequency domain using Discrete Cosine Transforms (DCT), quantised and variable length encoded for transmission. For predicted blocks, a suitable patch of reference pixels is located, then the *difference* between current and reference data is transformed, quantised and encoded. H.264/AVC also employs intra-prediction, where intra macroblocks are predicted from the edge pixels of available neighbouring macroblocks.

In H.264/AVC, the integer-based DCT transform is an approximation of the DCT transform. It is very simple and can be computed using only additions, subtractions and

Figure 2.4: The coefficients of a 4x4 discrete cosine transform in the spatial domain

bit-shifts. It is integer-based to avoid rounding errors and consequent drift between encoder and decoder. The algorithm for a 4x4 DCT transform is included in Appendix A.

Figure 2.4 is a visualisation of the 4x4 DCT in the spatial domain. A 4x4 DCT contains 16 coefficients and this is a greyscale visualisation of setting each single coefficient to its highest value and all the others to 0. This representation is somewhat different to the idea of edges and colours that normally dominates when considering groups of pixels. It is, however, a closer representation of how groups of pixels are processed by compression codecs.

Quantisation is the coarsest method of rate control in video compression and takes place in the frequency domain. Quantisation is performed as in Equation 2.1 where δ is DCT coefficients of a macroblock or residual, C is the compressed coefficients and Q_s represents the quantisation step as indexed by the quantisation parameter (QP) [31]:

$$C = round(\frac{\delta}{Q_s}) \tag{2.1}$$

A high QP yields a smaller bitstream at the expense of more compression artifacts and lower quality. Higher QP indexes larger Q_s and means more frequency coefficients are rounded out entirely. In H.264/AVC, the range of QP is 0 (lossless) to 52. Crucially, with a suitable rate control algorithm, QP varies both spatially and temporally throughout a video sequence. Neighbouring macroblock syntax elements can include a change in QP. Thus an object's visual quality can also vary spatially and temporally. Unlike natural changes in an object's appearance, changes due to video compression quality may be measured objectively using data from the compressed bitstream, however this only applies to the most recent compression. Should a bitstream be recompressed, then evidence of previous compression details must be derived from the pixels themselves.

2.1.2 Encoder Options

Although the industrial standards define many aspects of video compression, there are a number of options which are left up to individual encoder implementations. Open source encoders such as x264 [30] often come with a wealth of optional settings. Each of these contributes to how an encoder compresses a specific sequence. These include:

- Rate control
- GOP structure
- Motion search
- Macroblock type selection
- Optional encoder modes such as H.264/AVC's deblocking filter

How an encoder selects these defines how well it operates. For example, a given rate control mechanism may allocate proportionally more bits to frames that will be used as reference frames. This leads to higher quality reference frames, yielding better motion estimation matches, more skipped macroblocks and an overall more efficient bitstream. A simple encoder implementation might omit specific macroblock or frame types, so that these are never used, but still produce a compliant bitstream. Two different encoders might produce two very different bitstreams which are both equally valid, meet the required bitrate specification and produce video of equal quality.

The interplay of all these options means that different encoder implementations need not process the same sequence identically. This may give rise to different encoders exhibiting different "fingerprints", in much the same way as different camera models exhibit different photoresponse non-uniformities (PRNUs). The VISION dataset [20] demonstrated how different online applications (YouTube and WhatsApp) compressed video in different ways. This resulted in a reduction in performance of a proposed camera identification algorithm. Compression clearly reduces the appearance of camera identifying sensor noise, but it remains unclear if it replaces this with any features that allow identification of the encoder. Intuitively, looking at individual encoder implementations, it seems obvious that all the different choices would leave an individual encoder fingerprint, but in order to ascertain if this is feasible, it is necessary to derive compression parameters directly from pixels themselves. It is possible that different encoder options result in different bitstreams but near identical reconstructed frames.

2.1.3 Rate Control

In video compression, there are two possible modes of rate control: constant quality and constant bitrate. Constant bitrate (CBR) is most common in the wild, but constant quality gives reasonable insight into the compression mechanisms and removes the subjectivity of individual encoder rate control algorithms. Although the standards [27] and [28] define the syntax elements of compression from the decoder side, they do not define how these interact in an encoder to produce a bitstream of constant bitrate.

One of the main parameters involved in rate control is the quantisation parameter. Many CBR algorithms simply define a process to appropriately select the QP for every macroblock in a sequence such that the number of bits in the encoded sequence conforms to bitrate requirements. With rate control enabled, it is possible to perceive changes in a completely static scene. This is due to the balance of bitrate requirements and image quality optimisation. Later frames or macroblocks may "correct" or improve upon the appearance of an object, thus increasing the object's fidelity to the original, uncompressed frame. For example, the bit rate requirements may be such that the encoder has to encode naturally larger key frames at a reduced quality to avoid peaks in the bitrate. This means that key frames will use a higher QP than predicted frames, and the lower QP used in predicted frames will allow static regions in the frame to improve in quality over time. The use of different QP for different macroblocks also has knock on effects for prediction: different motion vectors may be selected on the basis of quality variation within the reference frame. Using constant quality mode removes a large proportion of this variation.

Constant quality is more commonly known as Variable Bitrate (VBR). VBR fixes QP to a defined level for every macroblock in the stream. This can be used to evaluate different encoder manipulations in isolation, or it can be used to gauge the complexity of video content itself. For example, flat regions of unchanging colour will make for a very simple video sequence which can be encoded using very few frequency coefficients and many skipped macroblocks. In VBR, this will result in a small number of bits. A natural scene with lots of subtle motion, say like a rippling river in the sunshine, or a tree with many leaves moving in the wind, will be very complex to encode and require a lot of syntax elements. In VBR, this will result in a higher number of bits. This method of operation is used for multi-pass encoding: the encoder itself is used to quantify how inherently complex a sequence is, then this knowledge is used to inform the rate control mechanism. Further examples of complex video content include camera zooms or through-frame motion where the scale of a given object is changing. Motion vectors and difference encoding are not well suited to large changes in object

Figure 2.5: YUV colour channels. Image is from (CASIA TIDE) V2.0. Left to right: Image, Y, U, V, (best viewed in colour)

scale or in plane rotation. The result of this is a larger VBR bitstream.

2.1.4 Encoder Evaluation

Encoders are often evaluated using full reference quality metrics such as Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity (SSIM) [36]. Rate distortion curves plot a full reference quality metric against bitrate so that different encoders can be compared. A number of YUV video sequences are publicly available [37] and these are used to assess different encoder options. In images or frames with a bit depth of 8, PSNR is calculated as:

$$PSNR = 20log_{10}(\frac{255}{\sqrt{MSE}}) \tag{2.2}$$

where MSE is the Mean Squared Error. PSNR does not account for the visual effect of neighbouring pixels. It is simply a measure of the difference between co-located pixels in the test and reference images. PSNR has many problems associated with it, the main one being that it does not necessarily represent human perception. Certain image processing techniques, such as Gaussian blur or speckle noise, have a disproportionately detrimental effect on PSNR values when compared to human perception. SSIM goes some way to addressing this, but the widely adopted gold standard in terms of image and video quality is simply user evaluation.

2.1.5 Colour Spaces

RGB is the most common colour space considered for image processing. For every pixel, a Red, Green and Blue value is defined. In the case of the commonly used RGB24, each sample is a single byte, so each pixel is represented by 24 bits, 8 bits red, 8 bits green and 8 bits blue.

Video compression typically uses the YUV colour space as a starting point. In this colour space, Y, the luminance channel, represents intensity. U and V are the chrominance or colour channels. Supplying only the Y channel gives a greyscale image. Human perception is much more sensitive to intensity than to colour or wavelengths [38]. This is because of the two different types of sensor in the human eye, both rods and cones detect intensity, but only cones detect colour. Compression methods take advantage of this fact by using sub-sampled YUV channels as the first stage in compression. YUV 4:2:0 is one of the most commonly used colour spaces [39]. In YUV 4:2:0, every square of four Y-channel pixels corresponds to a single U and a single V value. This means that the two colour channels together occupy half the size of the intensity channel. Thus the total pixel information is halved prior to the application of standard compression techniques, and this loss is seldom perceived. This colour space is often used as the starting point for standard encoder evaluation, and much of the available unprocessed video content in [37] uses this colour space. Figure 2.5 shows how a colour image can be decomposed into the Y, U and V channels.

YUV 4:2:2 is a similar colour space, where every square of four pixels corresponds to two U- and two V- values. YUV 4:4:4 is the full, uncompressed colour space. Separating channels into luminance and chrominance channels also allows different compression parameters to be selected for luma and chroma.

8 bits per sample is used throughout this thesis. Although H.264/AVC allows for more bits per pixel, most digital video available online is only 8 bits per pixel. The YUV colour space is used throughout.

2.1.6 Artifacts

There are a number of different artifacts and effects that can be observed in videos that are distinct from image data. Some of these are the result of compression, either resulting from bitrate constraints or a combination of encoder choices. Resizing or otherwise transforming compressed data may still exhibit block artifacts but they, too, will be resized. A number of these artifacts are listed here to illustrate how compressed frames can differ from compressed images.

Interlacing

One of the most visible distinctions between video and image data is when video capture uses fields rather than frames. This is a historical technique where all even

numbered lines are captured at one time and all odd numbered lines are captured at another time. Typically the field rate is double the frame rate. The fields are then interlaced to convert to frames.

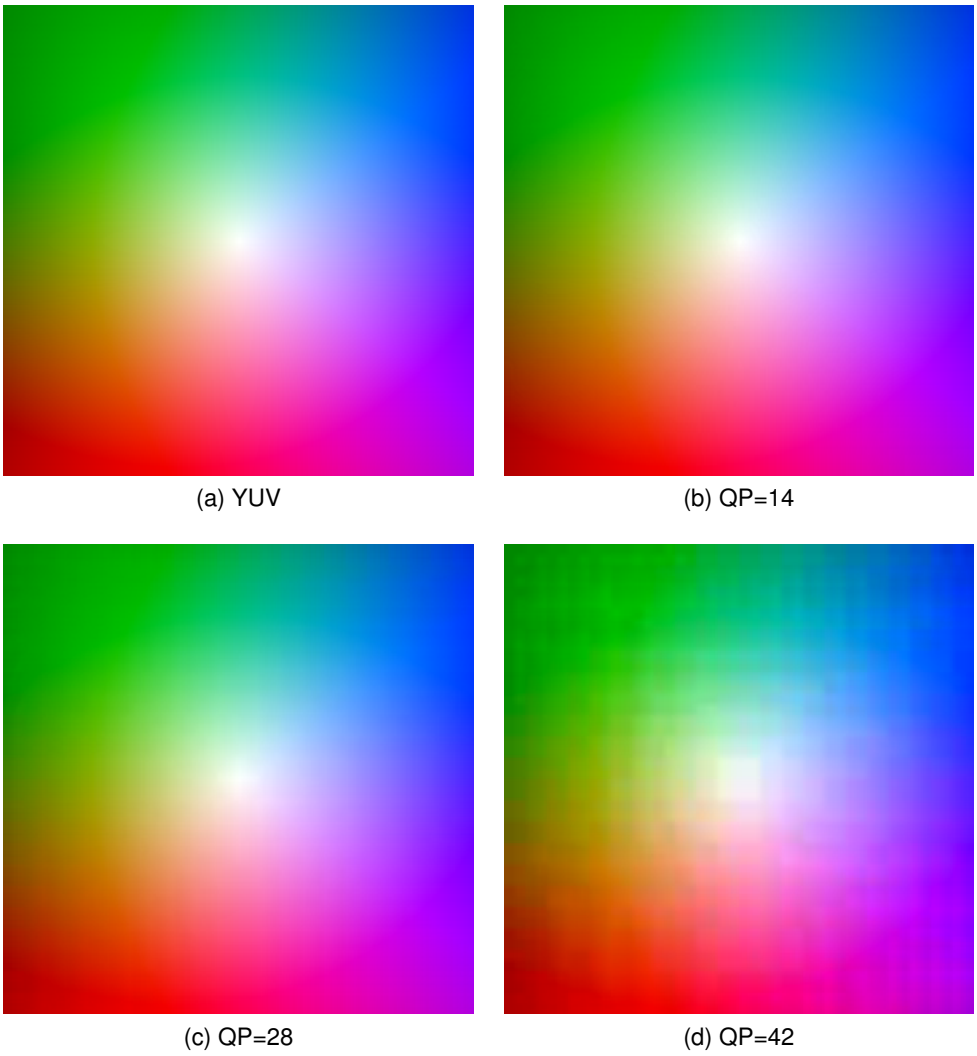
Figure 2.6: An example of field encoding. The fringe effects can be clearly perceived around the edges of the person (sequence is "Handball2" from VOT 2016 [40])

Visually, field encoding manifests as "fringes" round the edges of objects (see Figure 2.6), and these can be reduced by a de-interlacing filter before the compression process and encoded as frames, or they may be coded as individual fields within the compression process itself. The standards [28] and [27] allow syntax specifically for field encoding, although this is no longer provided in more recent standards [41]. Field encoding is relatively common in the video domain, particularly in broadcast sports footage, and is not typically seen in single images unless they have been extracted from a video sequence. Interlacing produces visible comb effects but enables further compression and general bitrate smoothing. Rather than encode a complete intra frame, only an intra field needs to be encoded, thus reducing any intra-frame related peaks in the the bitrate. It is possible to encode a sequence as fields even if the sequence is captured as frames. In such a case, the combing effects would be absent. A de-interlacing filter allows combing artifacts to be interpreted by human viewers as motion blur. One of the effects of offset fields is to create errors in localising object edges.

Blockiness and Banding

The most commonly observed artifacts in compression are blockiness and banding. Both of these can be observed easily at high QP. Blockiness can be attributed to the fact that both H.264/AVC and MPEG-2 are block-based codecs which use macroblock units with edges parallel to the edges of the frame. The step change between macroblock content manifests as visible horizontal and vertical lines. In intra frames, these are equally spaced throughout the frame, and can be particularly noticable in content

with smooth transitions, as can be seen in Figure 2.7



(a) YUV (b) QP=14

(c) QP=28 (d) QP=42

Figure 2.7: Higher QP leads to increased blockiness. There are differences between uncompressed YUV and QP=14 but they are nearly invisible to the naked eye. The blockiness becomes apparent to the trained eye at QP=28 and is obvious by QP=42.

Banding is also due to the step change between macroblocks. Smooth colour transitions often occur in otherwise flat, edgeless frame regions, like sky. Individual macroblocks are quantised, filtering out some of the high frequency components of the smooth colour transition. Neighbouring macroblocks show a different colour and there is therefore an unsubtle edge between macroblocks. As predicted macroblocks utilise reference regions that overlap more than one macroblock of the reference frame, block edge artifacts become banding. Figure 2.8 shows an example of this. H.264/AVC's deblocking filter is designed to counter this by filtering at block edges, however the filters limitations mean that it can only be applied horizontally and vertically, not diagonally.

(a) YUV (b) QP=42

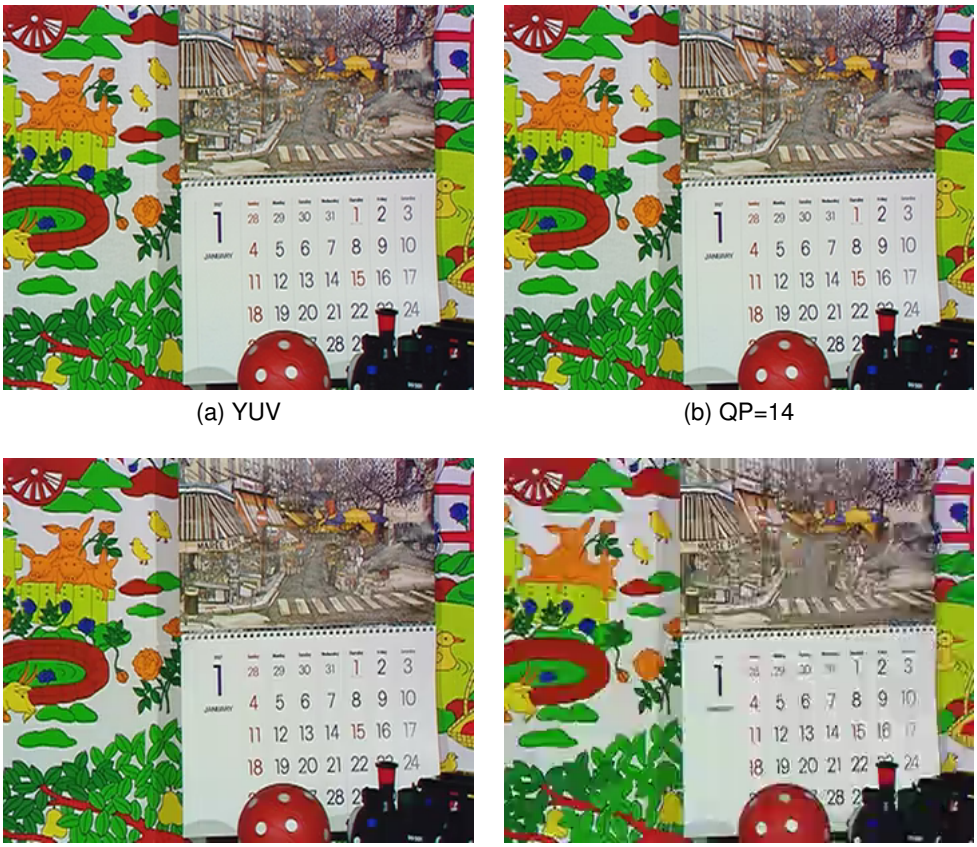
Figure 2.8: An example of banding. High QP has lead to lower quality overall, but banding is visible in the blue sky region at the top of the image.

Figure 2.9 shows the effect of QP on natural content. There are no visible differences between lossless compression and QP=14. Fine details, such as the faint vertical lines between dates on the calendar, are lost between QP=14 and QP=28. Above QP=28, further details are lost and some DCT coefficient noise is visible when the image is enlarged.

Compression artifacts can often be reduced by reducing QP. In VBR mode, QP reduction is a simple task, however in CBR mode it is a fine art. CBR relies on an encoder's internal rate control mechanism. Encoders may allocate fewer bits to "flat" regions in order to preserve detail in other places, and this results in more banding. Noise filtering original content will reduce the complexity of the sequence and, with a fixed bitrate constraint, allow more bits to be allocated to encoding content rather than noise.

Temporal Upsampling

Although techniques exist for the production of temporally upsampled video (see Section 3.2.2), videos still exist where temporal upsampling is achieved simply by repeating frames. This may happen automatically, for example, if a 25 fps video is re-encoded as a 30 fps video. The effects on visual quality are negligible, and often go unnoticed by viewers. Unfortunately, compression adds noise to upsampled frames. As can be seen in Figure 2.3, an encoder compresses the difference between the *reference frame* and the incoming frame. With two identical frames, this should, theoretically result in an entire frame consisting only of skipped macroblocks. Reconstructed frames in lossy compression, however, are not bitwise identical to the original frame, so there may be some differences to encode. Moreover, continuously updating rate control



(a) YUV (b) QP=14

(c) QP=28 (d) QP=42

Figure 2.9: Higher QP leads to poorer quality on natural content. The differences between uncompressed content and QP=14 are nearly invisible. At QP=28, the vertical lines on the calendar start to be filtered out by the quantisation process. The content compressed at QP=42, although still recognisable, suffers from visible low quality. GIFs of these sequences can be viewed on the author's GitHub.

options may result in the two frames being compressed with different quantisation parameters. Error concealment functionality within the encoder may cause some macroblocks to be encoded as intra type. Therefore, a sequence of two identical frames will not necessarily result in a bitstream consisting only of skipped macroblocks. The net result of this is that an uncompressed sequence of identical frames may be compressed into a sequence of non-identical frames.

Figure 2.10 shows an example of a repeated frame in a compressed sequence. The visual difference between the two frames is minimal and the binarised difference image shows a distinctive sparse pattern with many squares. These squares may indicate macroblock or sub-macroblock differences, but the analysis of this particular image is further complicated by the fact that the VOT2016 dataset [40] is distributed as folders containing JPEG images of each frame in the sequence. These particular JPEGs are

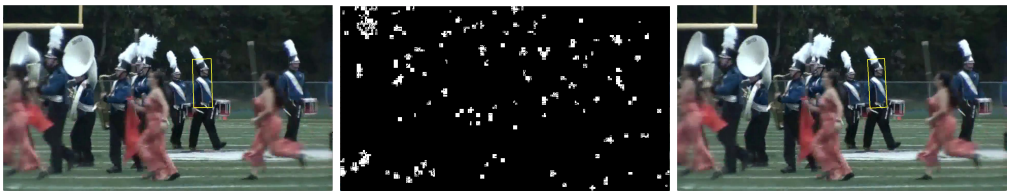


Figure 2.10: An example of a repeated frame due to temporal upsampling. The binarised difference frame shows the two frames are not bitwise identical due to minor differences in the pixels attributable to the video compression process, yet they are visually identical. (Sequence and bounding box annotation from VOT 2016 [40], sequence is "Marching", frames 2 and 3)

compressed at 75%, and any differences between frames may be magnified by this. This gives a good example of how compression can cause difficulties in video analysis, and a similar application is given in Section 3.5.

2.2 Deep Neural Networks

The deep neural network is a machine learning technique with many applications which has recently become one of the forerunners in computer vision research. Where previous image classification techniques involved hand-crafting single features, and then labouriously combining these to form a classifier, deep neural networks have replaced this with methods to derive useful classification features directly from data itself. Deep neural networks are a rapidly developing field of research, particularly in the domain of supervised learning. Convolutional neural networks have recently come to the fore in image and video analysis. Advances in computing power, and, in particular, GPUs have helped to accelerate their development. Due to the data-hungry nature of DNNs, some research efforts have been directed towards gathering and processing/labelling large datasets such as ImageNet [14], Microsoft COCO [42], YouTube-8M [43]. More recently, image datasets have been engineered for specific purposes like rebroadcast detection [2], source identification [20] and tampered image detection [44, 45].

The idea of neural networks is not a new one. The original idea of using neurons to build a network came about in 1940s. The main advances, however, did not come about until sufficient computing power was available. These days, there are many different network architectures designed for supervised image classification alone. The seminal LeNet-5 [46] learned to recognise the digits of MNIST [47] using only 7 layers: 3 convolutional layers, 2 trainable sub-sampling layers and 2 fully connected layers. An error rate of less than 1% was reported. AlexNet [13] was a network trained on 1000 classes of ImageNet [14], reporting an error rate of 15.3%. The network used 5

convolutional layers, 2 maxpooling layers and 2 fully connected layers. AlexNet also used ReLU activations which helped to train the many layered network. Following on from these network architectures, VGG-Net [48] used only 3x3 kernels in its convolutional layers, but utilised many layers. The Inception module [49] introduced the ability for network layers to contain multiple different-sized kernels, in order to allow the network to learn which kernel size was most suited to the data through the use of a 1x1 kernel. Training deeper neural networks led to the vanishing gradient problem, and Residual Networks or ResNets [50] were designed with skip layers to help mitigate this effect. It has even been shown that computer image classification now surpasses human abilities [51].

Networks are not just for classification, however. Generative Adversarial Networks (GANs) were designed to create images. In GAN architecture, a generator network learns to create an image from a noise vector. A discriminator network then learns to discriminate between generator-created samples and real samples from a dataset. The two networks train together, ultimately improving the generated images until they are indistinguishable from the real dataset. GANs have seen a lot of development for generating images, and form an important part of digital image manipulation, as can be seen in Section 3.2.

The authors of [52] showed that CNNs with sufficiently large numbers of parameters can fit to **any** training dataset, even randomised labels. These CNNs may not necessarily generalise to different test sets, regardless of techniques designed to prevent overfitting such as data augmentation and drop out.

Although DNNs are versatile, and achieve outstanding results in classification of visual data, they are difficult to explain. The generally proposed idea in CNNs for image classification is pixels into edges, edges into shapes and shapes into objects. Recent research, however, has shown that this is not necessarily the case, but that engineering a training set specifically to bias towards shape rather than texture can achieve improvements in classification accuracy [53]. Fergus and Zeiler [54] also provided a visualisation of features relevant to different CNN layers for image classification tasks. There has yet to be significant work in understanding how the gaps in the human visual system have indirectly influenced learning in neural networks through the mechanism of digital image and video compression.

For completeness, the maths behind the training of a fully connected deep neural network is discussed in Appendix B.

2.2.1 Network Parameters and Hyperparameters

When constructing a DNN for a particular application, there are a number of choices that have to be made. The network architecture, hyperparameters, data preparation/augmentation, training time all must be considered. It can be said that traditional feature engineering has been replaced with dataset and hyperparameter engineering. Although there are no hard and fast rules for these decisions, recommendations can be taken from the literature and improvements made after trials.

Different types of network layer can be combined to form many different network architectures and several different architectures are used in this work. Table 2.3 defines how these layers are abbreviated when the networks are defined in the relevant sections.

Table 2.3: Network layer abbreviations

Abbreviation	Meaning
fc-*m*	Fully connected layer with *m* nodes
conv*n* x *n*-*m*	Convolutional layer with kernel size *n* x *n* and *m* nodes
maxpool*n* x *n*	Pooling layer (maxpool) with *n* x *n* kernel (stride indicated separately)
softmax	softmax layer as described in Appendix B

Convolutional layer kernels are usually square in image applications and tend to cover an odd number of pixels such that there is a central pixel and the resultant transform does not skew the data. Kernels with even dimensions are rarer but do have their place, particularly in applications which are looking for sub-visible patterns in pixels which may not be related to the visible objects in the image [55, 56, 57]. In Chapter 5, different kernel dimensions are examined.

Alongside the basic architecture of a neural network, there are hyperparameters that also help to control training. These are summarised in the following paragraphs.

The size of the mini-batches is an important consideration when training a neural network. Training with a very small batch size causes network weights to fluctuate a lot, and training may result in oscillation which never converges. Training with a large batch size can result in very long training times.

Learning rate controls the magnitude of the effect that the loss has on the weights of the neural network. Learning rate lies between 0 and 1.0. A large learning rate means that the model converges very quickly but the converged model may be sub-optimal. With very low learning rates, each gradient descent step is very small, meaning that the model trains slowly with many small steps, but is more likely to find an optimal

solution. It is common for models to train using a learning schedule where the learning rate starts off high to facilitate rapid learning but decreases after a given number of mini-batches or epochs, to bias towards an optimal solution.

Drop out is where some neurons in each layer can be disregarded, or dropped out, temporarily during training. Drop out helps to prevent over reliance on a single feature. It also enhances model generalisation and helps to stop overfitting. Overfitting is where the model simply memorises the training data, and the learning does not generalise to any other data from the dataset. It has been shown that networks with sufficiently large numbers of parameters can overfit on any dataset, and even achieve high accuracy on a dataset where the labels are completely randomised [52]. Drop out was trialled in some of the DNNs used in this thesis.

2.2.2 Data

The ideal labelled dataset for supervised learning should be truly representative of the real problem and the individual feature vectors of the dataset should be uncorrelated. Torralba and Effros [58] showed that large image datasets, such as [14, 59], are seldom completely uncorrelated, but the size of the dataset goes some way to compensating for this.

A dataset is often partitioned into training data, test data and validation data, and the three sets should be completely disjoint to avoid data leakage. Training data is supplied as inputs to the network in order to adjust the network parameters to minimise the loss. The validation set provides a means to objectively measure the accuracy of the network on data not in the training set while tuning the hyperparameters. The test set is for testing how well the network generalises to new data. The biggest threat to model accuracy is overfitting and care must be taken during training to avoid overfitting, by using dropout or early termination. A key indication of overfitting is where training loss is still decreasing, but accuracy on the test or validation set is also decreasing. It is possible to overtune hyperparameters on the validation set and create a network with high accuracy on the training and validation sets which does not generalise to the test set. Validation sets are sometimes omitted if hyperparameters are not subject to overtuning. Some datasets are already partitioned into test/train/validation sets, but others can be manually partitioned or analysed using cross-fold validation.

Although large datasets exist, they can be enlarged even further using data augmentation. In the case of image classification, this can involve slightly different crops of an image, left-to-right flipping amongst other options. In this work, we do not use data

augmentation when training networks to identify compression parameters, as it has a high probability of effecting evidence of compression in the pixels themselves. Some data augmentation is used in Chapter 4 to train an image classifier.

In CNN image processing, images are cropped or scaled to a fixed dimension and whitened before being used as input to a network. In this work, because compression features are of fixed size, scaling the images is avoided, however cropping and whitening are used. Details on whitening are in Appendix B.1.

Many of the datasets in this thesis are synthesised from uncompressed content (Chapters 4, 5, 6). Some publicly available datasets are used (Chapter 7). The datasets used in each of the individual sections of this thesis are detailed locally.

2.2.3 Evaluation Metrics

For fully supervised classification problems, networks can be evaluated using metrics that compare the number of correct predictions with the number of incorrect predictions. This turns the evaluation of fully supervised networks into a binary problem: the predictions are either correct or incorrect. In image classification tasks with many different classes, top-N accuracy is also used, where a prediction is considered correct if the correct label appears in the top N predictions by the network.

One of the simplest evaluation metrics is accuracy, defined as in Equation 2.3.

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \tag{2.3}$$

With a perfectly balanced dataset, where there is the same number of samples in every class, accuracy is a reasonable metric. Problems with accuracy come with imbalanced datasets. Given a binary dataset with 99 examples of the positive class, a classifier can achieve 99% accuracy by simply labelling everything as the positive class. Since many of the datasets used in this thesis are synthesised and created with balanced classes, accuracy is often the metric of choice.

In some binary datasets, classes are imbalanced. This is a particular problem in the field of video tampering localisation where tampered content may make up only a small portion of the complete sequence. Authentic pixels usually outnumber manipulated pixels. For imbalanced binary datasets such as these, F1 score (Equation 2.4) and Matthews Correlation Coefficient (MCC in Equation 2.5) can be used. In Equations 2.4 and 2.5, TP means "True Positives", TN means "True Negatives", FP means "False

Positives" and FN means "False Negatives". A "True Positive/Negative" is where the predicted label matches the actual class.

$$F1 = \frac{2TP}{2TP + FN + FP} \tag{2.4}$$

F1 score takes both precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) into account. It ranges from 0 to 1, where 1 represents a perfect result.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.5}$$

MCC provides a score between -1 and 1 where 0 represents uncorrelated data, 1 is completely correlated data and -1 is completely inversely correlated data. This is particularly useful for when classes can be completely flipped, in the case where there are two classes to differentiate between but the actual label is less relevant than the distinction between the two classes. Both F1 and MCC are still subject to negative effects from class imbalance and this is accounted for in Chapter 7.

2.3 Conclusion

Video compression is almost unavoidable in digital society. The many different compression codecs are designed with the human visual system in mind, and utilise many different modes and techniques to meet bitrate and filesize requirements, while maintaining visual quality as far as possible. On top of the numerous different industrial standards for compression, there are many different implementations, and within each implementation, many different ways to compress a single sequence. Video compression encoders can utilise many different options, and therefore detecting compression parameters is a multi-variable problem. Individual encoders may leave a digital fingerprint on the pixels themselves, but detection of this fingerprint is not necessarily trivial. Although the effects of compression remain largely beneath the threshold of human perception, they are not beyond detection by deep neural networks.

Deep neural networks themselves also comprise many parameters, and the selection of these parameters influences how well they work. From network architecture, to hyper parameter selection, to data processing, DNNs are a multi-variable solution themselves and best practices are often gleaned from related work.

Chapter 3

Video Tampering Techniques

The majority of online video is compressed by well-defined industrial standards. *Some* online video is tampered or manipulated in some way. In this chapter we present a novel investigation current video tampering techniques. We examine a variety of available video tampering techniques and note how this has increased substantially in recent years. Video manipulation is currently a very active area of research and as methods which produce realistic video improve, it becomes vital to develop detection techniques which are tampering-technique agnostic. When human eyes fail to detect tampering, machine eyes must be well equipped to take over.

We also examine how new methods of video manipulation are evaluated, and show how human evaluation remains fundamental to this. We collate the current provision of available tampered video and highlight some of the challenges associated with existing datasets and how these might be avoided in future.

The main findings of this chapter were published in “Digital Investigation” in March 2019, [60].

3.1 Overview

The synthesis of convincing fake video content has increased recently due to the development of intelligent models [61, 62, 63]. Selective modification of image content has been possible for some years, but the application of similar techniques to video has been too labour intensive to see mass use. If each frame in a video is treated as an independent image, there are simply too many images to process efficiently. This has changed with increased computing power and the advent of deep neural

networks. Deep learning techniques have seen great success in many applications recently. Generative Adversarial Networks (GANs) in particular have been used to alter source video: to re-enact human facial expressions [64], change the weather [65] and to apply face-swapping [66]. Human facial re-enactment is a relatively new but common area of research where a simple, talking head is visually altered to mimic the facial expressions of a second actor [64, 6, 17] or to match a different audio track [67, 68]. This may have innocent applications, such as re-dubbing a film in a different language or creating new movie scenes using old footage of an iconic actor, but it can also be used to produce convincing fake content. In some circumstances, fake content is convincing enough to reliably fool human eyes. The authors of [17] even found that human viewers performed little better than random guessing when trying to ascertain whether facial re-enactment footage was authentic or synthesised. A deep neural network, however, could distinguish between the authentic and forged footage with ease.

Research into data-driven machine learning has also prompted the gathering of large image and video datasets such as ImageNet [59], Youtube-8m [43] and CelebA [69]. These datasets are a valuable resource for further research into convincing image and video forgery and in some cases, [17], a library of resources available for use in tampered datasets. The influence of these datasets has led to an increase in the application of deep neural networks to tampering. Spatially localised changes in video footage, such as face swapping, can change the entire context of a news story or film and can have repercussions for the people portrayed. Already, videos which have been cleverly edited to change the context of what was said by influential people have gone viral [1]. If that can be done with unsophisticated editing techniques, it is worth considering what more could be achieved with modern techniques.

There are already a number of recent surveys which review tampering detection methods [21, 70, 71, 72, 73]. Tampering detection methods are broadly categorised as active or passive, with more focus on passive tampering detection methods. This is practical, given that active tampering detection relies on advanced preparation of video. A review of passive tampering detection in video is provided in [71, 73] and, more recently, [21]. Inter-frame tampering detection is specifically covered in [70]. Pandey et al [72] cover tampering detection through noise in images as well as video. There are, however, far fewer reviews on tampering itself. As the number of tampering techniques increases, it becomes vital that these are reviewed independently in order to explicitly collate those tampering techniques that have no corresponding effective detection

[1] "Video of Barack Obama speech circulating on the Internet was edited to change his meaning": https://www.politifact.com/truth-o-meter/statements/2014/jun/23/chain-email/video-barack-obama-speech-circulating-internet-was/ Accessed 2019-1-24

methods. The work in [74] provides an overview of personation, specifically how a person's likeness in appearance and voice can be forged in videos either physically or digitally. However it is important to objectively catalogue current known techniques that can be used for video tampering in order that they can be identified and, ultimately, detected or countered. Many detection techniques are explicitly tailored to specific tampering methods. For example, the authors of [17] trained a deep neural network to detect their own video content changes in order to assess the quality of their content-altering techniques; [75, 76] focused on inter-frame tampering; [77] created tampered sequences using established in-painting techniques [78, 79] to assess their detector. All of these techniques worked well, but all of them required prior knowledge of the *type* of tampering, and further work showed that the techniques could not be guaranteed to generalise [80]. As tampering methods multiply, it becomes important to fully assess new detection methods, and to appreciate which types of tampering techniques they can feasibly detect and which they are blind to.

Wang and Farid [81] noted that, at the time of their 2007 publication, there were very few video tampering detection techniques. This is no longer the case, however, many published techniques, specific to particular types of tampering or source authentication, were assessed on proprietary datasets which remain unreleased [76, 82, 83, 84, 85]. In some cases, [86], datasets are detailed sufficiently in the literature so that they can be exactly replicated, provided the sequences used for dataset synthesis are available. This serves to evidence the fragmentation of the tampering detection field. In reality, a tampered video may be subject to a variety techniques, including combinations. For tampering detection to be effective, individual detectors must be analysed and matched with an appropriate type of tampering. In order for that to happen, we must review and differentiate types of tampering.

In this chapter, we catalogue and analyse the current trends in digital video manipulation techniques from simple edits to fully synthetic video. This provides motivation for work towards methods of universal video tampering detection. To prepare for this work, we list currently available tampered datasets and identify challenges associated with them. We thoroughly examine problems in dataset gathering and dissemination, specifically including challenges created by compression.

3.2 A Spectrum of Manipulation

The most convincing lie has its roots in truth and, arguably, the most convincing falsified video content is an altered original rather than entirely generated content. However, video manipulation is not a single process and instead can be arranged into a spectrum of techniques from least to most complex and invasive.

In [21] video tampering was defined as "a process of malicious alteration of video content, so as to conceal an object, an event or change the meaning conveyed by the imagery in the video". Similarly, [87] described image forgery as "the digital manipulation of pictures with the aim of distorting some information in these images". Here, video tampering is regarded as any technique which is intended to produce manipulated, photo-realistic content using authentic sources. There is no defined limit as to when authentic video becomes tampered video, only a forensic history of video processing. Similarly, malicious intent is difficult to quantify, and so tampering detection and video authentication techniques must focus on forensic analysis, providing objective localisation of inconsistencies within digital footage that may imply content alteration.

It is important to note that here, we only examine *digital* video tampering: video content can also be "staged" whereby the video file is an authentic record of events, but the events themselves were contrived or unnatural during filming. Detection of staged video involving natural ballistic trajectories is examined in [88], and [74] details how plausible audiovisual personation is achieved in front of the camera during filming. Digital video forgery can take a number of forms [21] and Figure 3.1 gives an overview of the classical interpretation.

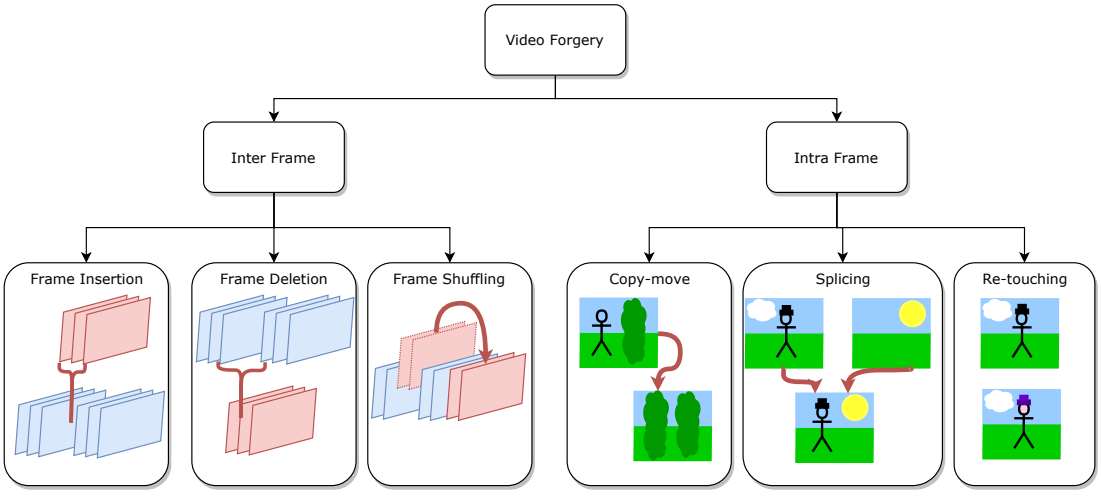


Figure 3.1: Traditional video forgery categories

In the past, video tampering methods have been simply classified as inter- or intra-frame [21, 70] (Figure 3.1). The terms inter- and intra- frame primarily distinguish temporal tampering from spatial tampering. Inter-frame tampering is performed on a sequence-level: the pixels of individual frames are unaltered, but the sequence as a whole is changed. Intra-frame tampering is performed on a pixel-level: some spatial regions are altered, but alterations temporally correlated to form a convincing forged region. The term "inter-video tampering" can also be used to describe the merging of content from two different videos [89]. Traditionally, this has been a form of splicing, where chroma-keyed objects taken from one sequence are inserted into another, as in [16]. Recent developments, however, mean that convincing synthetic regions [64, 66, 7] or even whole videos [62] can be synthesised automatically from authentic content. This development means that we must now consider different levels and categories of video tampering. It is important to be aware of the different categories because video tampering is designed to be invisible to human eyes, and detection techniques often address only one type of tampering.

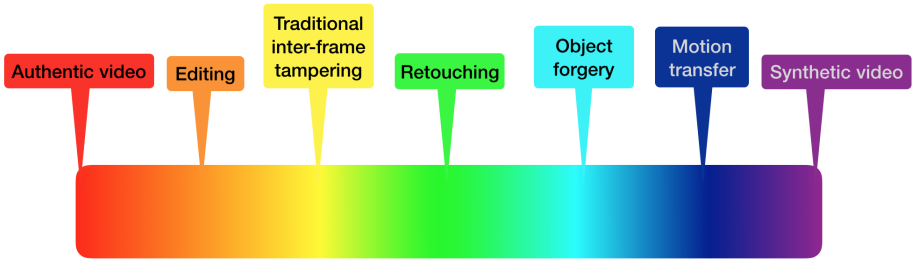


Figure 3.2: Video tampering spectrum

The current field of video tampering may be viewed as a spectrum, as in Figure 3.2, where different types of video tampering are ordered according to potential to deviate from authentic source. Whereas the traditional view in Figure 3.1 provides only two categories of video tampering, the spectrum in Figure 3.2 demonstrates that there are now multiple ways to produce convincing, falsified content. This distinction is important because detection methods often address one particular type of video tampering such as object forgery or inter-frame tampering. In [21] tampering detection methods are categorised as recompression, inter-frame forgery or region tampering detection. With current tampering techniques, the distinction is less clear cut. Moreover, multiple tampering techniques can be applied to the same sequence.

Figure 3.2 summarises the current categories of video tampering. Video editing compiles single camera shots into full films complete with scene cuts. Although clever editing may change the context of a video, scene cuts are not usually deliberately concealed. Video clips of maliciously edited content exist in mainstream media and are

surprisingly effective at disseminating misinformation through social media. Traditional inter-frame tampering, where edits are concealed, may reorder events or even remove or insert events into the timeline, but its content-altering effects are self-limiting. Retouching temporally or spatially upscaled content, or applying global filters to improve perceptual quality may affect every pixel in a video sequence and can cosmetically alter content. Retouching can also be applied to specific regions. Intra-frame tampering and other object forgeries such as inpainting can alter content and context, as can motion transfer. Finally, fully synthetic video or synthetic regions can be produced. Unlike historical animations, the synthetic content of today looks convincingly realistic. The following subsections 3.2.1 to 3.2.5 detail examples from each of these types of video manipulation.

Table 3.1 shows how motion transfer and video synthesis techniques have become common in recent years and demonstrates how methods of evaluation remain relatively underdeveloped. Evaluation techniques are difficult to define since there is no pre-defined ground truth for tampered video data. Every new method can be assessed qualitatively. Methods which seek to imitate authentic video, such as frame interpolation, can use full reference quality metrics such as SSIM and PSNR. As can be seen in Table 3.1, video manipulation methods use user studies to evaluate their output or simply publish examples of their methods for future evaluation. However, even user studies can vary. Some ask users to classify frames as tampered or authentic. Some request a user preference between the published method and other, similar methods. In a related field, image inpainting evaluation techniques are reviewed in [90] and these can all be used to assess the spatial features of inpainted video or indeed, any form of tampering which affects individual frames. No-reference video quality assessment is a large and open field and although we do not cover this here, we point to this field to at least partially inform on tampered video evaluation.

3.2.1 Editing and Inter-frame Tampering

Editing and inter-frame tampering both change the order of the frames in the video without changing the contents of each frame. In the case of editing, the goal is to turn a series of single camera shots into a coherent story. Clever edits can be used to turn innocent footage into propaganda,[2] but scene cuts are not hidden and such videos are not above suspicion. In inter-frame tampering, the goal is to *invisibly* remove, re-order or alter events.

[2] "Israeli army edits video of Palestinian medic its troops shot dead to misleadingly show she was 'human shield' for Hamas", The Independent, https://www.independent.co.uk/news/world/middle-east/gaza-protests-latest-idf-condemned-edited-video-angel-of-mercy-medic-razan-al-najjar-a8389611.html

Table 3.1: Video tampering and evaluation methods: Qual= qualitative analysis; PSNR=Peak Signal to Noise Ratio; SSIM=Structural SIMilarity; UP=User preference to previous methods; UR=User comparison with real video; Rel=Released Sequences

Reference	Year	Type of Tampering	Qual	PSNR/ SSIM	UP	UR	Rel	Other
ETS [79]	2004	inpainting	✓					
Ha et al [91]	2004	frame interp.	✓	PSNR				
Patwardhan et al [78]	2007	inpainting via temporal copy-move	✓					
Wexler et al [92]	2007	inpainting, frame interp.	✓				✓	
Shih et al [93]	2011	object forgery	✓				✓	
SULFA forged [1]	2012	object forgery	✓				✓	detection
SULFA supplemental [94]	2013	object forgery	✓				✓	detection
Newson et al [95]	2014	inpainting	✓				✓	
Ardizzone and Mazzola [89]	2015	copy-move	✓				✓	
Ebdelli et al [96]	2015	inpainting	✓	PSNR			✓	
Lotter et al [97]	2015	frame pred.	✓					error
Dar and Bruckstein [98]	2015	frame interp.	✓	PSNR				
Face2Face [6]	2016	motion trans.	✓				✓	
Le et al [61]	2017	inpainting	✓				✓	
Suwajanakorn et al [64]	2017	motion trans.	✓					
Liu et al [65]	2017	style trans.	✓				✓	
Niklaus et al [99]	2017	frame interp.	✓	PSNR				
MoCoGAN [100]	2017	motion trans.	✓			✓		ACD
Walker et al [101]	2017	frame pred.	✓					Inception
FaceForensics [17]	2018	motion trans.	✓			✓	✓	detection
Recycle-GAN [63]	2018	video synth.	✓		✓			✓
Wang et al [62]	2018	video synth. (sketch)	✓		✓			
Chan et al [7]	2018	motion trans.	✓	SSIM				LPIPS
Jiang et al [10]	2018	video synth. (blurred image)	✓	PSNR				
Wang et al [102]	2018	video synth. (smile)	✓		✓			
Xiong et al [103]	2018	video synth. (time-lapse)	✓		✓	✓		
Babaeizadeh et al [104]	2018	frame pred.	✓	✓				
Zhao et al [105]	2018	frame pred.	✓	PSNR	✓			ACD
SCGAN [106]	2018	video synth. (human pose)	✓		✓			pose eval.
SDC-Net [107]	2018	frame pred.	✓	✓				
Cai et al [108]	2018	frame pred./interp.	✓	✓				Inception

Edits in inter-frame tampering are deliberately concealed so as to be invisible to the human eye. The invisible scene cut is also a technique used for artistic effect to give the impression of a single shot. It can be seen, or, rather *not* seen in some music videos such as "Wannabe" by the Spice Girls. Detection of visible scene cuts in video has been studied extensively so that key frames can be identified for efficient compression and or used to condense/index the sequence [109]. Invisible scene cuts are studied in the context of inter-frame tampering detection [110, 111, 112].

Such is the theoretical simplicity of generating an inter-frame tampered sequence, that many tampering detection methods, such as [75, 76, 113, 114] generate their own datasets from single-camera video sequences such as SULFA [1] or Derf's media collection [37] or even film their own sequences as in [85]. SULFA [1] replicates single camera sequences as obtained from CCTV footage, and, therefore may be representative of the most likely application of inter-frame tampering: altering CCTV evidence. Derf's media collection [37], on the other hand, provides publically available uncompressed sequences and allows researchers complete control over the forensic history of synthesised tampered sequences [114].

It remains unclear how widespread inter-frame tampering is in the wild because, if it is done correctly, it will be undetectable by human eyes and remain above suspicion. Meanwhile, it is important that synthesised datasets are as high quality as possible. In creation of inter-frame tampered datasets, [85, 86, 115] simply removed pre-determined frame numbers from each sequence, and it is unclear if this caused visible effects, thereby reducing the problem to simple cut-scene detection. In [114] frame addition and removal was limited to the beginning of each sequence, but again it is unclear if the additions were visible. Simply reversing the sequence from the point of tampering may effectively locally conceal the edit but would make the detection of the edit much more challenging. Recent developments in video quality assessment mean that temporal glitches in video can be objectively quantified [116] and also smoothed [117] to achieve temporal consistency. Future datasets for inter-frame tampering can use this to improve such that inter-frame tampering techniques can be deployed in the wild.

As noted in the review in [21], many inter-frame tampering detection methods suffer from limitations which are often related to consistencies within the dataset that may not translate to other video data. These consistencies are often related to video compression. The authors of [118] note that some tampering detection techniques are tied into the fixed Group of Pictures (GOP) size, commonly used in MPEG2 [28] to minimise error accumulation due to non-integer frequency domain transforms. Later video compression standards, such as H.264/AVC [27], use integer-based transforms so error

accumulation drift between encoder and decoder is no longer an issue, and therefore key frames are used only as access points into the stream or efficient compression of visible cut scenes. Moreover, sequences compressed using [27] may no longer exhibit visible evidence of key frames. Since intra-frames generate more bits, however, it is common to compress them using a marginally higher quantisation parameter than the surrounding predicted frames in order to smooth the bitrate.

Although inter-frame tampering and frame deletion detection is widely studied in the literature [21, 76, 119], effects similar to inter-frame tampering can be achieved using a spatio-temporal copy-move. Rather than replacing complete frames in the sequence, only partial frames containing motion or objects to be concealed are replaced. With a static camera and consistent lighting, this is visually effective, and video edits prove near invisible to the naked eye. Indeed, some sequences which initially look like inter-frame tampering [94] are actually spatio-temporal copy-moves, as can be revealed by examining pixel-by-pixel difference between the authentic and tampered sequences (see Figure 3.3d).

3.2.2 Retouching and Resampling

Retouching involves adjusting pixels within an image using transforms or filters applied to the pixels themselves which may only have a low-level interpretation of video content. As the name suggests, retouching is less invasive to content than other types of forgery but may still change the context of a video. Moreover, retouching can be used on tampered video specifically as an anti-forensic device.

A retouching function R can affect specific pixels according to a binary mask, M:

$$V_{retouched} = R(M \odot V_{original}) + ((I - M) \odot V_{original}) \tag{3.1}$$

Here, I represents a matrix of ones and all matrices have equal size. Retouching can also be applied globally as in:

$$V_{retouched} = R(V_{original}) \tag{3.2}$$

Colour correction methods such as those available in Adobe After Effects may normalise lighting in a sequence of shots taken on different days under different weather conditions to create a convincing narrative. Similarly, colour grading can also be used to add effects or make video filmed during daylight hours appear to have been filmed

during twilight. A typical colour correction model works by adjusting the histogram of colour over a specified region, however the authors of [120] found that gamma correction (a form of colour correction) was particularly difficult to detect using a deep neural network. Where median filtering and Gaussian blurring could be detected reliably with over 91% accuracy, detection of gamma correction was only 57.6% .

Compression is often a necessary part of video processing but it can also be used as an anti-forensic method. Video compression standards such as [28, 27] reduce video file size but individual encoder implementations do not necessarily have an explicit understanding of video content. Compression has been found to reduce the efficacy of tampering detectors [17, 94, 115, 121] and has also been shown to reduce the classification accuracy of convolutional neural network (CNN) based classifiers [24]. The authors of [17] found that video compression [27] reduced the accuracy of deep neural networks trained to detect human facial re-enactment performed by [6]. Of seven different forgery detectors tested, the Xception network [122] was the most robust against compression, achieving 87.81% accuracy, compared with 99.93% accuracy on uncompressed sequences. Other methods [18] performed less well for forgery detectors on the compressed dataset with performance for some [123] dropping as low as 55.77%. This may be attributed to the depth of the Xception network. The authors of [121] also account for compression in their SYSU-OBJFORG dataset. In benchmarking it using seven common steganographic features, they, too, found a drop in accuracy on the reduced bitrate video. While their ensemble-based detector achieved precisions in the range 78.9-93.15%, this reduced to 61.85-79.34% when bitrate of the video data was halved. Halving height and width of the video sequences also reduced precision to the range 73.02-90.28%, which was not as significant as bitrate reduction. In detection of frame deletion, it was found in [115] that an SVM conditioned on uncompressed data to detect dropped frames did not perform well on compressed data taken from YouTube, with accuracy dropping below 37%. It is clear from this that standard video compression can reduce some of the machine learned features associated with tampering. This is further examined in Chapter 7

Compression artifact removal is another example of retouching and methods such as [124, 125, 126] have been applied to JPEG images. The authors of [127] used a deep residual network to reduce artifacts in a BPG (Better Portable Graphics) [41] compressed frame. More recently compression artifacts have been removed in the video domain [128], where videos compressed using HEVC and H.264/AVC were retouched to increase Peak Signal to Noise Ratio (PSNR). PSNR is defined as in section 2.1.4.

Although [128] achieved overall improvement in PSNR, it was unclear if this resulted in a gain in perceptual quality at specific bit rates. Given that some tampering detection

methods, such as [129], actively utilise compression structures, methods which alter the underlying patterns of compression in video frames could be used as anti-forensics in the future.

Artificially upscaling video frame size [11], frame rate [8, 130, 131] or bitrate can be a form of video tampering. High quality video content is more desirable to consumers, and larger file sizes for the same film/footage are often indicative of higher quality, with bitrate often taking the place of quality in common parlance. Compression encoders utilise a specified bitrate, even if this means compressing existing compression artifacts. Bitrate upscaling can be done innocently as researchers seek to provide high quality, "uncompressed" datasets and either overlook or deliberately replicate compression artifacts in the pixels of mined data.

Artificially increasing the spatial dimensions of video has been commonly done historically as Standard Definition (SD) content is displayed on High Definition (HD) screens. More recently, super-resolution has evolved from an image enhancement technique to use within videos [11, 132, 9], and the metrics commonly used for evaluation are, again, PSNR and SSIM: full reference quality metrics. It is important for spatially upscaled video to demonstrate temporal coherence. The authors of [11, 132], also assessed the temporal coherence of their super-resolution sequences using a technique called "temporal profile". This is where single rows of pixels are viewed along with their temporal neighbours from different frames, and temporal inconsistencies or video "flicker" shows up as hard edges in the resultant image. The work in [116] is also a method of assessing temporal consistency.

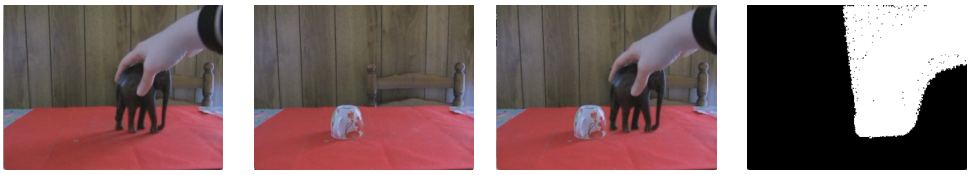
Video deblurring [133] is another example of retouching and a dataset exists to facilitate the development of this [134]. The dataset was filmed using a GoPro camera at 240fps and then downsampled and blurred so that a non-blurred ground truth can be supplied for each blurred frame, thus enabling deblurrers to be assessed using full reference quality measures such as PSNR. Super slow motion has recently become a strong field of research with many new techniques for temporally upsampling video [99, 8, 135] to create a slow motion effect in the absence of a high speed camera. Previously, upsampled video would simply involve frame repetition or averaging. The field of motion compensated interpolation improved upon this [91, 98, 99] so that interpolated frames were less obvious. The work in [91] is an early example of motion compensated interpolation, and the authors used block-based motion estimation similar to that used in video compression [28] to inform interpolation and create sophisticated intermediate frames. The frame rate upconversion algorithm was objectively assessed by downscaling some publicly available uncompressed sequences and then comparing the original sequence with the computed upscaled version using PSNR. In

[98], the authors showed how their work in frame rate upscaling could be used to improve low bitrate video compression. In [99], a CNN was used to interpolate between frames. The authors obtained their training data from high quality YouTube channels and downsampled from 1080p to 720p in order to reduce the effects of compression. A user study confirmed that their interpolated frames were better than previous state-of-the-art. Objective assessment of results used sampled alternate frames from a popular YouTube video and used PSNR to compare interpolated frames with actual frames. In [8], multiple frames were synthesised between two authentic frames using a CNN trained on high frame-rate (720p, 240fps) video from YouTube and a high frame-rate dataset [136]. The synthesised frames were assessed using high frame-rate video and it was found that PSNR between interpolated frames and ground truth was higher than the previous state-of-the-art. As noted in [137], inpainting or video completion (Section 3.2.3) can also be used to resample a video, and entire frames inpainted.

Retouching might be one step of many in tampering, and although it does not necessarily alter context, it can be used to make tampering detection much more difficult. Countermeasures for anti-forensics are well studied in the literature [120, 18, 138], and datasets can be generated easily. In [18], a CNN was used to classify an image in terms of its anti-forensic processing. The labels used were: original (no processing), Gaussian blurring, additive white noise, median filtering and resampling. The CNN accurately detected the presence of each process over with over 98% accuracy using only the green colour channel. A new type of convolutional layer was designed to prevent the network from learning typical image features. The authors of [138] showed how their median filter detector could be used to localise median filtering within a spliced image and hence localise image tampering. Although retouching does not always correlate with tampering, localised retouching can be a strong indicator of splicing or other object forgery.

3.2.3 Intra-frame Tampering

Intra-frame tampering is where spatial content of individual frames is changed, that is, individual objects are added or concealed/removed. Intra-frame tampering is also known as “region tampering” [73] and applies equally to video and still images, although the video application is more complex. Care must be taken to ensure that spatial tampering across individual frames is coherent and does not cause visual jarring in the video. Intra-frame tampering methods in images were classified as spatio-temporal copy-move, splicing and retouching in [139], but here we discuss retouching



(a) Auth. 34 (b) Auth. 236 (c) Tamp. 236 (d) Difference

Figure 3.3: An intra-frame tampering example from [94]. 3.3a and 3.3b show authentic content. 3.3c shows the spatio-temporal copy-move and 3.3d shows the difference between 3.3b and 3.3c

separately (Section 3.2.2).

A spatio-temporal copy move can be defined by:

$$V_t^{Lj} = ((I - M) \odot V_o^{Lj}) + (M \odot V_o^{Lk}) \tag{3.3}$$

where I is the matrix of ones, M is a binary mask to localise tampering, V_o^{Lj} is an authentic sequence of L frames starting on frame j, V_o^{Lk} is the same sequence but starting on frame k where $j \neq k$. The frames/mask can be re-aligned or cropped so that any object or region of pixels from any spatial or temporal location can be copied to any location. V_t^{Lj} is a tampered video sequence:

$$V_t^{Lj} = [v^j, ..., v^{j+L-1}] \tag{3.4}$$

In spatio-temporal copy-move attacks, all the data used in the video forgery V_t comes from within the same video sequence V_o. For example, complete objects from frame k in the sequence are inserted into frame j using mask M. Figure 3.3 shows an example. This is similar to image-based copy-move where the pixels involved in the tampered region come from within the image itself. Although this reduces the range of potential content, it helps to minimise differences between legitimate and tampered regions. There is less need to alter the colour histogram or adjust the frame rate to make tampered content consistent with authentic content if both share the same source. Using a copy-move attack, objects can be added to a sequence by adding foreground objects, or concealed/removed from a sequence by duplicating background regions from within the same frame or from within a different frame in the same sequence.

Some versions of copy-move attacks simply duplicate a still background region, [1], and these can be detected with relative ease by high coherence or abnormally low

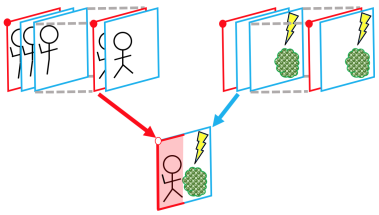


Figure 3.4: Forging a video (best viewed in colour), Red borders/dots indicate key frames in the sequence. Blue borders (no dots) indicate predicted frames. The hybrid frame is shaded red where the pixels have come from a key frame, and unshaded where the donor frame was a predicted frame

motion within the tampered region, [94]. Other methods [94] duplicate an entire spatio-temporal region, and this is more difficult to detect. Although duplicates can be detected by matching copied region to original data, this becomes more difficult in the presence of compression [94]. A copy-move attack can be detected in images by identifying and locating duplicated regions, and this has been done using search based on brute-force pixel matching, region matching or key point matching [139, 140]. While this type of copy-move detection is feasible in images, video adds another dimension and searches become an order of magnitude more complex. Previous video inpainting attempts such as Temporal Copy-Paste (TCP), where identical pixels are used frame after frame to conceal an object within a video [1], or Exemplar-based Texture Sythesis (ETS) [79] were detected by [77] where detection was based on correlation between adjacent frames which was either too strong or not strong enough to be authentic video.

Splicing is an extension of spatio-temporal copy-move. In a splicing attack, two sets of pixels from different sources are combined as in Figure 3.4. The sources can be videos or even still images (as shown in Figure 3.5 where the spliced object is a still picture on the wall). Equation 3.5 defines splicing:

$$V_t^{Lj} = ((I - M) \odot V_{s1}^{Lj}) + P(M \odot V_{s2}^{Lk}) \tag{3.5}$$

Where video sequences are defined as in Equations 3.3 and 3.4, $s1$ means sequence 1, $s2$ means sequence 2 and P is an optional processing step which can be applied to aid blending between different source videos. The frames/masks of the two sources can be re-aligned or cropped so that any object or region of pixels from any location from source 2 can be pasted into any location in source 1.

Splicing has large potential for context-changing edits because two entirely different subjects can be spliced together. Any source sequences involved can be retouched

(a) authentic (b) tampered

(c) authentic (d) tampered (e) mask

Figure 3.5: An example of spliced content. 3.5a and 3.5b are from VTD [15] and the spliced content (a picture on the wall) comes from a static image. 3.5c, 3.5d and 3.5e come from D'Avino et al [16] where the spliced content comes from a chroma-keyed video

(see Section 3.2.2) using colour correction or temporal synchronisation before or after a video splice in order to visually camouflage spliced content, or even to launder the splicing operation to make it undetectable to existing forensic tools.

Copy-move and splicing are also known as "object forgery" [121] because they often involve removing or adding complete objects to videos. Introduction of an object to a video can be done using chroma-keying techniques, as in the field of video special effects [141]. Chroma-keying requires filming against a single colour background under specific lighting conditions to facilitate segmentation of foreground objects. An example of object forgery using chroma-keyed sources is shown in Figure 3.5d. Other segmentation methods such as [142, 143] may be used in place of chroma-keying so that foreground objects can come from any sequence without the need for special green-screen filming. The authors of [142] applied segmentation to the optical flow of videos in order to distinguish foreground and background objects in sequences with moving cameras. In [143], segmentation was achieved using supervoxels, using spatiotemporal uniformity in pixels to group them into voxels and supervoxels to represent different objects in the sequence. Masks produced by [142, 143] could be used in place of specially filmed green-screen sequences, thus rendering any video susceptible to use in object forgery.

Inpainting or video completion [78, 92, 95, 96, 137] allows removal of objects from an image by interpolating remaining pixels to conceal a "hole" left by a removed object or

corrupt section of video. It is useful for error concealment when streaming video over an unreliable channel and can be used to restore old film, but it can also be used to deliberately remove objects or even frames from a sequence. Video in-painting techniques were surveyed recently in [137], where it was noted that many methods of video in-painting rely on patch completion where the missing spatio-temporal volume is filled using small patches from within the same video sequence, thus reducing inpainting to a particularly complex spatio-temporal copy-move. This is evident in early video in-painting methods such as [78, 92]. In [78], static background and dynamic foreground were assumed, thus highlighting one of the challenges associated with video completion: motion. This was handled by first registering or aligning the frames of the sequence. Background mosaics of the video sequence were then constructed by removing all non-static objects, and foreground mosaics contained all the objects moving relative to the camera. Missing data was then inpainted by finding close matches in the mosaics and interpolation with texture synthesis between the matched segments. The authors of [92] used space-time volumes of 5x5x5 pixels taken from other areas of video sequences to fill in the space left behind be a removed object. Motion was accounted for by representing each pixel not only in terms of its RGB components but also two components based on the derivation along the x-, y- and t- dimensions. This method also allowed temporal and spatial upsizing as spatio-temporal holes had varying dimensions in the spatial and temporal axes. More recently, the work in [95] realigned source patches to create closer matches and less warping of synthesised video content. Initial values for missing pixel data were also explicitly defined in [95].

The assessment of inpainting quality in the image domain was critically reviewed in [90], and user survey to assess the visibility of inpainted regions remains the gold standard. The authors also noted that Video Inpainting Quality Assessment remains an important, open area of research. Indeed, in the absence of an accepted video completion quality assessment, authors [61, 95, 96] simply publish videos of their inpainting techniques applied to standard sequences online and [137] notes this as a trend. The authors of [96] also provided their original sequences along with defined masks so that future inpainting techniques can be applied to precisely the same data for comparison. The existence of these inpainted sequences provides a good source of data for video tampering detection research.

In recent times, advanced image processing inpainting techniques [144, 145] have evolved. Image interpolation methods vary but can leave behind distinctive artifacts such as fish scale and checkerboard artifacts. However, in [145], these patterns had been significantly reduced. A reduction in spurious artifacts mean that these image

inpainting methods may be more suitable for transferral to video completion. Alternatively, it may be that applying these techniques to video allows for the reduction of these artifacts through use of temporal data for filtering.

Inpainting can be used in conjuction with spatio-temporal copy move to create complex forgeries. An early example of this can be found in [93] where the authors changed the winner of a 100m race. The authors considered the video as a series of layers. They applied in-painting using unoccluded areas of background and interpolated/sampled the motion of forged runners to make them move slower/faster relative to other objects in the video. While individual frames taken from the forged sequences looked visually convincing, full video sequences are not currently available for full analysis. Indeed, assessment of tampered video remains an open problem, one which the authors of [93] suggest is best tackled by forgery detection methods. Although the subject matter of this video was somewhat ambitious for its time, and the authors explicitly target the field of video special effects, it gives a good idea of how tampering can be used to court controversy.

3.2.4 Style and Motion Transfer

Style transfer is a new method of image and video manipulation which has been facilitated by the advent of Generative Adversarial Nets (GANs), which were first established in [146] and extended to conditional GANs in [147]. Style transfer can completely change the context of an image or the subject of a video. It is strongly related to motion transfer because the resultant video is a combination of motion from one source video and content or subjects from another. Combining the two can be viewed as a style transfer when the style of the content source is mapped to the motion source or it can be considered motion transfer when motion is mapped to the content source.

Examples of style transfer in the image domain include [148] where features from one object are mapped to a similar object: a scene can be changed from a summer scene to a winter scene; a horse can be exchanged for a zebra [149]; Google Street View House Numbers can be translated into MNIST-style digits [65] and evaluated using accuracy on a CNN trained to classify MNIST. Examples in the video domain include motion transfer [64] as well as style transfer.

An example of a conditional GAN used to perform style transfer can be found in the seminal Pix2Pix [148], which performs image to image translation. A GAN consists of a generator network and a discriminator network. In Pix2Pix, the generator network maps an observed image and a random noise vector to a generated image. The

discriminator network then uses both the mapped image and the observed image to classify the mapped image as an example from the authentic dataset or one from the generator. Authentic examples given to the discriminator dictate the "style". This architecture is distinct from a non-conditional GAN where the discriminator network sees only the mapped image. The authors of Pix2Pix noted that they could achieve very good results based on small datasets of only 400 authentic images and so the GAN can be trained for a multitude of applications. For example, the input image can be a sketch or a semantic segmentation mask and the mapped image can be photorealistic, or vice versa; daytime scenes can be mapped to night; the mapping process can even perform inpainting or background removal. The versatility of [148] has also spawned further applications in the video domain including [63, 100].

Motion transfer is similar to style transfer where the motion of one object is passed on to another object. Early applications were mostly specific to human facial re-enactment such as lip synchronisations and expression translation between talking heads [64, 6]. Thies et al [6] presented the first real-time facial re-enactment system that used only RGB as input. The method used authentic frames from a target video and transformed them to match the facial expressions and mouth motions from a source video. Face2Face did not use a GAN, but instead used facial tracking to estimate the position of the source and destination faces and interpolate the expressions between them. The manipulated faces were therefore the equivalent of "new", unprocessed content. In [64], the authors added video re-timing for realistic head motion to fit the context of the spoken word and used a recurrent neural network (RNN) trained on hours of footage of the particular subject to transform an audio track into mouth shapes. While [64] was not real time, and required many hours of video footage to train the RNN, it was capable of producing a representative video from audio and stock footage, whereas [6] required video for both source and target. More recently, motion transfer has been achieved using models based on style transfer.

MoCoGAN [100], used GANs in a similar way to Pix2Pix [148]. Content and motion were treated independently in MoCoGAN and video sequences expressed as:

$$Z_i = Z_c \times Z_m \tag{3.6}$$

Every frame in Z_i has a content vector, Z_c, and a motion vector, Z_m, associated with it. In order to perform motion transfer, the content of one sequence was substituted with the content of another. The architecture consisted of two distinct discriminator networks: one to classify real and generated images (or frames), and one to distinguish real and generated video. The video discriminator was responsible for smooth video

generation. Similarly, there were two connected generator networks: one to generate motion, the output of which was used to condition the content generator which produced video frames. The motion generator network was a recurrent neural network (RNN) which modelled motion through time. Motion content could also be extracted from a different sequence and hence motion can be transferred between two similar videos. The authors of [63] also applied motion transfer to videos, successfully replicating lip motions. Because [63, 100] are both based on style transfer, they can also be used to create photo-realistic synthetic video from semantic segmentation masks (Section 3.2.5).

The assessment of GAN-produced images and videos remains an open problem. In [65, 148], translated images were objectively assessed using the accuracy of CNNs pre-trained on authentic images in the output-style classifying the translated images. It was found that the CNNs classified the translated image of [65] with more than 90% accuracy. Image translation methods from [65] were also applied to some street driving video sequences, and qualitative analysis of the results showed a convincing, low frame rate video where the weather had been translated from sunny to snowy or the lighting mapped from day to night. The authors of [150] applied neural style transfer to photographed objects spliced into images of paintings, thus reducing the visibility of the tampered object. A user study found that their edited image set achieved similar user scores to an unedited image set meaning people could not reliably localise such processed image edits. Although [65, 148, 149, 150] show a method to alter image content, they do not assess whether there is a counter method which can detect these alterations. Since all of these methods employ the use of GANs, it is implicit that there already exists a network which has been trained to discriminate between authentic examples of the style and synthesised content, but due to GAN convergence, this network may not be optimal for detection. In the video domain, the authors of [64, 6] have released examples of their work to the public. In [67], the authors used a user study to compare their audio-to-video speech synthesiser to both a previous model and a motion capture solution. The study showed that their work advanced the state-of-the-art as their examples were preferrable to human eyes when compared to previous speech synthesis, but not when compared to motion capture generated video. The authors of [17] performed a user study on their tampered dataset and found that, when asked to differentiate between tampered and authentic videos, humans achieved no better than random guessing. Generic motion re-enactment and video generation has also been studied in [100] however state-of-the-art is not yet of a standard where such tampered videos are high quality content.

Image to image style translation can be applied to video frames to universally change

the overall context of the video. Complimentary work can be found in [117] where the authors examined the removal of flicker from a sequence of frames. They specifically aimed to allow the use of image style transfer on individual frames to produce a temporally coherent video sequence, independent of the style transfer method. In [7] the authors used style translation on videos to synthesise video content of people performing dance moves they had never done. Pose estimation was used as an intermediate step. A conditional GAN was trained to map a stick-man pose estimation to a photo-realistic video frame using the previous frame to condition the GAN. They then applied a spatio-temporal smoothing to generate convincing videos that showed a target actor dancing in a manner defined by a source actor from a different video. The video sequences were assessed by extracting a pose from the mapped sequence and comparing it to the pose used to generate the sequence, and manual qualitative analysis of temporal qualities including some publicly released sequences. The authors conceded that there were still a number of challenges to overcome in this field, such as loose clothing and cluttered background, but it is easy to see that motion transfer can already be convincingly applied to human faces and bodies.

3.2.5 Photo-realistic Synthetic Video

Although purely synthetic video in the form of animation has been around for a long time, more recently synthetic video has been generated which is so photo-realistic that it could be mistaken for authentic, filmed content. In this section, we examine the most recent techniques in photo-realistic video synthesis and discuss their evaluation. Although video synthesis is not explicitly tampering an existing video, full convincing, photo-realistic video synthesis has the potential to be just as damaging as motion transfer or inpainting. It is important to examine it with a view to detecting it as a future research direction. Current trends in top international conferences on computer vision show that video frame prediction is a strong trend.

A short video sequence was extrapolated from the motion blur of a single image in [10]. The authors noted that the main challenge of this is temporal ordering. While the central frame of the synthesised sequence corresponds to the de-blurred image, the motion of individual objects in frames before and after is ambiguous. The authors proposed a pair-wise ordering invariant loss to aid convergence of their CNN, which was based on pairs of frames at an equal temporal distance from the middle frame. Although the de-blurring aspect of the technique improved on the previous method [134] for moderate blur, evaluation of the short synthetic sequences proved difficult. The ambiguity in temporal ordering could be resolved when the process is constrained to

temporal super-resolution. Video generation from a single image was also covered in [102] where Wang et al detailed a method to produce a short photo-realistic video of a smile from a single aligned face image. The method used a series of conditional Long Short Term Memories (LSTMs) to produce a sequence of facial landmarks moving from a neutral expression to a smile. A network similar to [148] was then used to translate the facial landmarks into a realistic video. A comparative user study found that the resulting sequences looked more realistic than a previous method, but the authors noted that it was difficult to evaluate such a method as there were no directly comparable existing methods. Both [10] and [102] extrapolated short synthetic video sequences based on a single image, and both noted challenges in evaluation. Xiong et al [103] produced short, realistic time-lapse videos of skyscapes, up to 32 frames from a single image using a two-stage GAN architecture. The first stage produced a sequence of frames and the second stage refined it to produce a coherent video. They also gathered a large dataset of real time-lapse videos from YouTube for the purposes of training. Again, there was no previous work available for direct comparison, but the authors were able to repurpose other network architectures to synthesise time-lapse videos. Evaluation was by user study where users were asked to identify the more realistic of two sequences. Although the proposed method outperformed all other synthetic videos, when comparing synthetic video with real video, only 16% of synthetic video tests were preferred to real video.

In [97], a CNN, LSTM and deconvolutional neural net were used together to predict the next frame in a video sequence. The authors noted that natural images were much more challenging than simple moving circle animations, and that, in predicting the next frame in a face rotation sequence, the network altered sufficient features so as to change the perceived identity of the face.CAPG-GAN [151] has recently been used to synthesise photo-realistic out of plane face rotations. The identity of the faces were well preserved. The work in [104] used a variational autoencoder to predict the next 10 frames from a 10 frame sequence. The authors tested their sequences on the dataset [152], among others, where they compared their predicted frames with ground truth frames using PSNR and SSIM. They conceded that assessing the quality of the predicted frames was difficult, and that the prediction yielding the worst PSNR was sometimes qualitatively the best result. They also publicly released many examples of predicted sequences. In [153], the authors used a two stream structure and RNN to perform frame prediction. The authors of [108] viewed frame prediction as analogous to frame interpolation and fully synthetic video generation. They successfully produced short video sequences which interpolated between two frames as well as predicting short sequences given only the first frame of each sequence. Evaluation used PSNR and SSIM for interpolated sequences and Inception scores for generated sequences

with no ground truth. Frame prediction was also covered in [105, 107, 154], and evaluation also involved full reference quality metrics. Although methods were evaluated using full reference quality metrics, there is no guarantee that frame *prediction* as opposed to interpolation will predict a frame that matches the original sequence, but it may yet produce a valid, realistic frame.

Some methods create synthetic video data specifically for machine learning datasets. In [155], a dataset of synthetic videos was created from motion capture data. The motion capture data was used to generate 3D models of human bodies which were combined with a texture map to add clothes and skin, and a static background image. The synthetic videos, rendered using Blender, were found to improve body part and foreground background segmentation. The Human3.6M dataset [152], includes some mixed reality videos which consist of a moving synthetic human model combined with a real video sequence. The real backgrounds included annotated occluding items so that synthetic human models could realistically interact with authentically filmed objects. Neither [155] nor [152] are specifically designed to fool human eyes, but instead intended to aid development of human pose estimation and body segmentation. Rather than annotate thousands of frames of authentic video, the synthetic human model is already annotated. This is an example of a non-malicious application of synthetic video, although the detection of the synthetic parts is often trivial to human eyes.

Wang et al [62] have already synthesised coherent, photo-realistic video sequences of up to 30 seconds from semantic segmentation mask or pose model sequences. A GAN was used, and a discriminator part of the GAN used to classify the content as an authentic video or not, similar to MoCoGAN [100]. Using a discriminator in this way ensured temporally coherent video. The authors conceded that significant changes in an object's appearance is still a substantial challenge and that their model was also prone to colour drift over time, however a user study showed that [62] produced video that was preferable to human eyes than that produced by MoCoGAN [100]. SCGAN [106] also performed a user study which put their synthetic video content at a higher level of realism than MoCoGAN. Recycle-GAN [63] was also used to generate photo-realistic video from semantic segmentation mask sequences and assessed their method's accuracy by asking users to classify videos as synthetic or real as well as comparison with existing state-of-the-art. Users were fooled into thinking synthetic video was real 28.3% of the time. Quantitative results were also obtained using the Viper dataset [156] which supplies pixel-level segmentation masks for computer game scenes with a high level of realism.

Methods of evaluation for synthetic video remains an open field. It can be seen in Table 3.1 that the main methods include full reference quality metrics PSNR and SSIM

as well as a variety of others such as Average Content Distance (ACD) [100, 105], Learned Perceptual Image Patch Similarity (LPIPS) [7] and tampering detection methods. Many of the techniques for synthetic video generation also utilise user surveys to assess the quality of synthetic video, and in many cases, evaluation is relative to previous related work (Table 3.1). It can be inferred from this that although current methods do not yet reliably generate video that is photo-realisitic enough to fool human eyes in all cases, improvements are continuous and incremental. It is simply a matter of time before photo-realistic synthetic video becomes mainstream. As [17] showed, some techniques are already indistinguishable from authentic video for human viewers. Similarly, the work in [157] also showed that human viewers are not particularly good at detecting tampering in images, achieving 46% accuracy on classifying a tampered image as tampered or authentic. Tampering techniques were one of: splicing, copy-move or object erasure. Participants had to classify the image as tampered or authentic and indicate the manipulated region of any image identified as tampered. Answers were considered correct only if the correct classification was given and, in the instance of tampered images, the correct region was identified. Human participants sometimes resorted to using context, such as identifying a particular person in the images, to infer tampering or not. The authors of DeepFakeVidTIMIT [158] even showed that deep neural network face recognition methods are also susceptible to video tampering. They created a dataset of swapped face videos using a GAN-based technique and tested to see if the swapped faces were accepted by two face recognition algorithms. The false acceptance rate was high with over 85% of low quality faces and over 94% of high quality faces accepted.

This raises the problem that, in future, not only will humans be unable to detect tampered or even photo-realistic synthetic video, they will also be blind to whatever tampering technique has been applied. Moreover, some machine vision methods will also be susceptible to tampered video, accepting it as authentic. When this is the case, universal tampering detection systems will be required to fill the gap, and these must be developed urgently if detection systems are to keep pace with tampering methods. For this, datasets are required.

3.3 Image Tampering Detection

To advance the field of universal video tampering detection, it is vital to gather datasets of independent examples of video tampering techniques. In this section, we look at the lessons relating to tampered image datasets that can be learned from the application of deep neural networks to the problem of tampering detection.

As machine learning techniques come to the fore in tampering detection, the collation of large datasets to train and test networks becomes desirable. However it is important to realise that *any* consistencies within labelled classes may be exploited as features by deep learning techniques, including any features arising during dataset generation that are unrelated to actual tampering. In 2011, Torralba and Efros [58] discussed how bias is ubiquitous within computer vision datasets. Images from the same dataset exhibit characteristics specific to that dataset, so much so that a basic support vector machine (SVM) classifier trained to label a given image with its associated dataset achieved reasonable accuracy of 39% over 12 datasets. Each dataset has its own inherent distribution which may be irrelevant to the real world situation, and may be overlooked by human eyes. This problem is subtly highlighted by the advance of deep learning, particularly in the field of image forensics.

A good example of unintentional features comes in the CASIA2 TIDE dataset [44]. This large dataset consists of 7491 authentic and 5123 tampered images which use splicing or copy-move techniques. The size of this dataset makes it an attractive option for deep learning and over 97% classification accuracy has been achieved by [159]. However, as noted in [160], compression applied to tampered images of the dataset differs from that applied to authentic images. Put simply, during dataset generation, tampered images were compressed twice, authentic images were compressed only once. There were also patterns in the colour space resolutions with tampered images more likely to have lower colour channel resolution. This means that classifying a CASIA2 image as tampered or authentic can be accurately achieved using features of compression, recompression and colour resolution. The recompression step may have arisen from the tools used to tamper the images, but it is independent of the tampering task itself. There is no reason that an authentic image cannot be innocently recompressed.

In [161], dataset weaknesses such as those in CASIA-2 were used as an explanation for the sharp drop off in CNN classification accuracy whenever the test images were compressed. Classification accuracy dropped from 97.44% on unprocessed CASIA-2 image patches to 68.11% when the images were compressed with JPEG quality factor 90, a fairly light compression. The authors proposed a means to circumvent this dataset flaw by extracting authentic *patches* from tampered images, however they did not report whether this reduced the drop-off in accuracy when the source dataset was compressed, nor did they report on a CNN *trained* using compressed image patches. Tampered and authentic patches may be extracted from only the tampered data but only if reliable localisation masks exist to differentiate tampered and authentic pixels. Although this indicates that the features learned in a deep neural network trained on

CASIA-2 will not transfer to other data, it may be possible that other datasets, and also tampered images in the wild, suffer from the same weaknesses. The fact remains, however, that maliciously tampered images are not necessarily recompressed, and authentic images are not necessarily compressed only once.

High levels of classification accuracy were also achieved by a deep neural network on the large rebroadcast dataset presented in [2]. This dataset comprises over 29000 images, half authentic and half rebroadcast in some way. Rebroadcast techniques included printing out and rescanning/photographing the images, screen grabs and screen photography. While some traditional techniques [162] demonstrated poor accuracy on this new dataset, a CNN trained on 60% of the images and tested on the remainder achieved over 97% accuracy. In this case, recompression is very likely a necessary feature of retransmission, so features that emerged during CNN training are a true reflection of the real process of retransmission. One way to objectively assess this is to check the performance on a rebroadcast test set gathered independently. If a deep neural network exploits unintentional weaknesses inherent in a particular dataset, then the learning will not transfer well to other, similar datasets unless they exhibit the same features.

Table 3.2: CNNs for image anti-forensics detection

Reference	Detection of:	Dataset	Accuracy
Bayar and Stamm [18]	Gaussian blurring, additive white noise, median filter, resampling	proprietary	99%
Choi et al [120]	all combinations of Gaussian blurring, Median filtering	[163], [164]	>91%
Choi et al [120]	Gamma correction	[163], [164]	57.6%
Amerini et al [165]	double compression level	[166]	83.5% - 99.9%
Boroumand and Fridrich [167]	low-pass-, high-pass-, denoising-filters and tonal sharpening	[163]	>95%
Agarwal et al [2]	rebroadcast	public [2]	>97%

Table 3.2 shows how CNNs excel in detection of image anti-forensics. A detection or classification accuracy of over 95% is a common occurrance. Anti-forensics are methods designed to “launder” tampering and thus fool tampering detectors. Laundering techniques include general filtering methods such as compression, median filter and Gaussian blur. This field is emerging rapidly because large datasets can be synthesised with relative ease, and this makes it particularly appropriate for machine learning. Datasets such as BOSSBase [163], UCID [166] and Dresden image

database [164] provide a large variety of unprocessed images to which known anti-forensic techniques can be universally applied and subsequently detected.

In [167], a CNN was trained to identify laundering techniques applied to an image. The laundering techniques, applied singly, were: low-pass-, high-pass- and denoising- filters and tonal sharpening. The authors first compressed the images of the dataset, then applied a single laundering technique and then rescaled, cropped and recompressed the resulting images. Compression was JPEG with a Quality Factor (QF) ranging from 75-95. They achieved over 95% accuracy in identification of the laundering technique used regardless of the image compression level, provided the CNN was trained on images with a QF similar to that of the test dataset. The idea that a training dataset must be well matched to the test data in terms of compression is also supported in [24] and is covered in more detail in Chapter 4.

The authors of [165] achieved 83.5% - 99.9% accuracy in predicting the most recent level of JPEG compression of a 64x64 image patch using a CNN. They found that the best results were achieved when the network was trained using both RGB inputs and also DCT coefficients. This method has the potential to be used as a means of detecting image splicing if two different images with substantially different compression parameters have been spliced together. It also has the advantage that training is performed on authentic or synthesised data only. Huh et al [168] also used only authentic data to train a tampered image detector. A deep neural network was trained on 128x128 pixel patches to identify 83 different attributes including: 80 EXIF image tags, plus Gaussian blurring, recompression with JPEG and rescaling. The resulting features were then used to train a Siamese Neural Network whereby patches coming from the same image were classed as a “match” and patches from different images were classed as a “no match”. This was then used to identify tampered images and localise where splicing had occurred. The method achieved very good rates of detection on tampered datasets which contained spliced images, but less well on datasets with other types of tampering, including copy-move.

All of these high accuracies show that machines are adept at detecting patterns in visual data which are invisible to humans. This makes designing a dataset which is representative of the problem of video tampering but immune to unintentional side effects especially challenging.

3.4 Tampered Video Datasets

With so many different methods of tampering already available, and the field progressing at an unprecedented rate, it is important for tampering detection techniques to keep pace. Unfortunately this is challenging because there are few large, diverse tampered video datasets. Pandey et al [72] noted that, at the time of their 2016 review, tampered video datasets lagged far behind tampered image datasets in terms of maturity. In this section, we examine existing video datasets and give recommendations for the design of new datasets. Table 3.3 provides a list of video tampering datasets, specifying the type of tampering applied and the size of the datasets. It can be seen that tampered video datasets are increasing in number and in size. Historically, tampered video datasets are created in the lab, but more recently videos have been collected "in the wild" with accurate labelling reliant on the publisher's honesty.

Table 3.3: Tampered video datasets

Name, Date, Ref.	Type of Tampering	Size	Details
SULFA forged 2012 [1]	spatio-temporal copy-move	5 tampered	Static camera, background duplicated to conceal objects. Part of a larger database including untampered video for camera identification
SULFA supplemental 2013 [94]	spatio-temporal copy-move	10 tampered	Duplicate spatio-temporal regions to conceal/introduce objects
Lin et al 2014 [77]	inpainting (TCP and ETS)	18 tampered x2	Only 4 sequences available for direct download
Newson et al 2014 [95]	inpainting	3 tampered	Masks supplied for two sequences, demonstration of inpainting rather than explicit tampering detection dataset
Ardizzone and Mazzola 2015 [89]	copy-move	160 sequences	Sequences synthesised from [1] and CANTATA datasets
VTD 2016 [15]	splicing; copy-move; frame-shuffling	26 tampered + related authentic	Distribution on YouTube means all videos affected by varying compression
SYSU-OBJFORG 2016 [121]	object forgery	100 authentic, 100 tampered	No source for public download
VISION 2017 [20]	source/social media platform identification	1914 sequences	648 straight-from-device videos, 622 YouTube and 644 WhatsApp. Future dataset extension planned via "MOSES" application [169]
D'Avino et al 2017 [16]	splicing	10 tampered	Binary masks provided, tampering is easily seen.
Le et al 2017 [61]	inpainting	53 sequences	Both object removal and object reconstruction, demonstration of inpainting rather than explicit tampering detection dataset
FaceForensics 2018 [17]	motion transfer ([6])	1004 tampered x2	Taken from Youtube-8m [43], one set self-reenactment, one set source-target translation
DeepFakeVidTIMIT [158]	deepfakes face swapping	640 sequences	Sequences from vidTIMIT [170], 16 pairs of similar people face swapped
Fake Faces in the Wild [80]	tampered video on YouTube	150 YouTube identifiers	Start and End times supplied for a temporal crop
Media Forensics Challenge [171]	tampered video	more than 4000 manipulated videos	Available on request, some parts reserved for the NIMBLE media forensics challenge
FaceForensics++ [22]	tampered video	1004 tampered x4	Extension of [17] with addition of FaceSwap and DeepFakes
Fake Video Corpus 2018 [172]	forged video in the wild	380 videos	Combines tampering techniques with misrepresented clips.

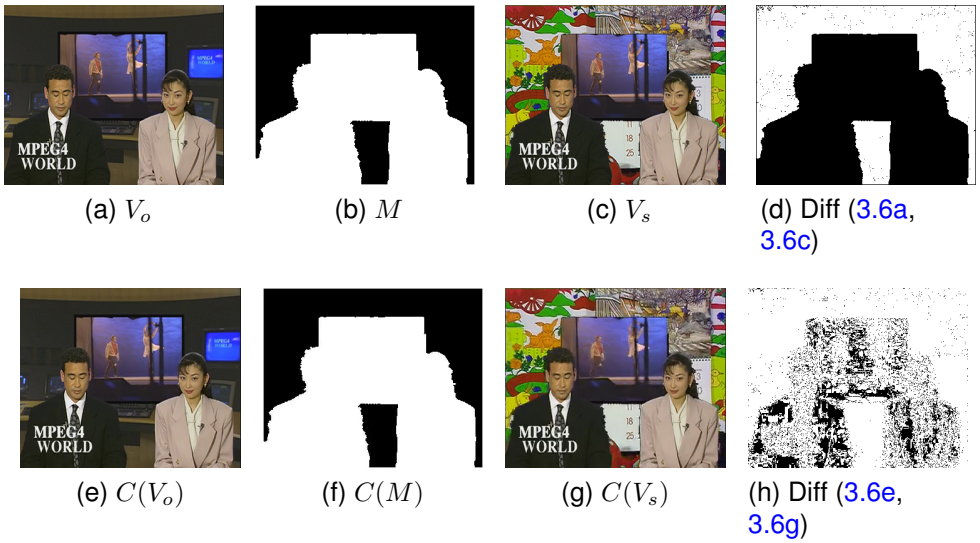
A number of tampered video datasets already exist, but these vary both in terms of processing and parameters. In [16] the authors supply tampered video along with an explicit pixel level binary mask detailing the chroma-keyed addition. Some video tampering datasets come complete with original and tampered videos, thus providing a means to calcualte all masks and labels associated with tampering [94, 15]. This allows for tampering detection and localisation in spatial and temporal domains. It also allows for any differences in distribution between tampered and authentic sequences to be overcome by extracting authentic patches from tampered sequences. However,

an accurate mask can only be extracted where videos can be synchronised and are identically processed post-tampering. Any recompression during distribution allows compression errors to creep in and increases the difficulty of extracting a bit-accurate tamper mask (as in Figure 3.6). Moreover, some pixels may be part of the tampered region but remain unchanged in value, and this makes for noisy masks in need of post processing.

Using differences between original and tampered videos may be inappropriate for temporally tampered videos [15], where a frame-by-frame label might provide more information. This can be achieved when unprocessed original and tampered sequences are provided. Indeed, public inter-frame tampered video datasets are in short supply, with many inter-frame tampering detectors simply building their own datasets from available sequences (see Section 3.2.1).

As can be seen in Table 3.3, many tampered video datasets focus on a single tampering method, such as splicing or object forgery or inpainting. Only the more recently compiled datasets such as VTD [15], FaceForensics++ [22] and Fake Video Corpus 2018 [172] demonstrate a variety of types. Fake Video Corpus 2018 [172] even goes further and includes video sequences which contain only authentic pixels but are misleadingly presented with captions that do not accurately describe the contents of the video. Variety is vital to accurately assess performance of video tampering detectors and support work towards universal video tampering detection. As discussed in [58], an approach using a combination of datasets will ensure more generalisable results with little need for specific domain adaptation. It is also important that tampered sequences are independent of tampering detection, so techniques such as [61, 89, 95] which publicly release their results are important to move forward both tampering AND tampering detection.

A number of datasets have produced and benchmarked with an existing detection technique [17, 77, 20, 22] and many achieve high levels of precision on their selected dataset, often over 90% accuracy. This tends to portray tampering detection as a solved problem on that particular dataset, which discourages researchers from publishing lower results. A tampering detection method based on motion residue was presented in [173], and the experimental dataset was gathered from several previous object forgery works [78, 93, 174]. Accuracy was lower than 90%. This contrasts with over 90% in [174] and over 99% benchmarked in uncompressed FaceForensics [17]. However, the authors of [80] showed that an Inception network trained on [17] achieved over 99% accuracy on the FaceForensics test set, but achieved less 9% accuracy on their “Fake Faces in the Wild” dataset gathered from YouTube. This clearly suggests



(a) V_o (b) M (c) V_s (d) Diff (3.6a, 3.6c)

(e) $C(V_o)$ (f) $C(M)$ (g) $C(V_s)$ (h) Diff (3.6e, 3.6g)

Figure 3.6: The problems with recompression in the distribution of tampered datasets: Left column shows uncompressed data, right column has been lightly compressed. Figs 3.6e, 3.6f, 3.6g are the compressed versions of 3.6a, 3.6b, 3.6c respectively. Figs 3.6d and 3.6h are both uncompressed and show binarised differences.

that FaceForensics exhibits some commonality in the tampered images which is specific to the dataset or tampering method.

In a real world situation, the method(s) of tampering will be unknown, and quite possibly invisible to human eyes. Moreover, some detection methods can be very specific to particular methods of tampering and fail all but completely on other types. Therefore, it is worth collating results on several different datasets such that work towards a universal tampering detector can be realised.

3.5 Dataset Dissemination

Methods of dataset dissemination are an important consideration as this can cause unintentional post-processing of video data. Although video sharing websites such as YouTube may seem like an attractive distribution option, [15, 80], any processing applied during publishing must be taken into account. It is possible to apply social media platform processing by uploading/downloading a video to/from a social media website, however the effects on the video are then irreversible. While researchers may add processing to an unprocessed video, they cannot remove it. Indeed, the effects of video processing by social media platforms on video are represented in isolation in a dataset

provided by [20] who found that sensor noise pattern used for camera source identification was adversely affected by processing on FaceBook, YouTube and WhatsApp, even when using high quality settings. These results are important in themselves as they show how tampering detection methods which rely on sensor noise, such as [82], can be defeated by virtue of the distribution platform alone. They also emphasise how post-processing can be easily overlooked. Similar problems may have contributed to the poor generalisation of deep neural networks reported in [80]. The training set, FaceForensics, is supplied directly as video files, unprocessed by YouTube and (optionally) uncompressed, while "Fake Faces in the Wild" consists solely of sequences from YouTube, all of which will be subject to compression, or recompression. With two such different forensic histories, it is unsurprising that a deep neural network method failed to transfer between the two datasets.

Figure 3.6 illustrates some of the complications associated with recompression. Starting with uncompressed data, a binary mask was created based on segmentation of static and non-static content, and two uncompressed sequences very simply spliced together. Figure 3.6d shows which pixels differ between Figures 3.6a and 3.6c and it can be seen that it is almost the perfect inverse of the mask (Fig 3.6b), with a few pixels that are identical between the original and spliced content. Figs 3.6e, 3.6f, 3.6g show the visual effects of compression on Figs 3.6a, 3.6b, 3.6c respectively. Figure 3.6h shows how compression has introduced tiny inaccuracies between the pixels of the original and spliced sequences so that the difference between them no longer provides a mostly accurate inverse of the mask. With some thresholding and morphological processing, the difference sequence could still be used to infer a mask, but the degree of accuracy suffers even under slight compression. A compressed mask, as shown in Fig 3.6f provides a more accurate ground truth than deriving the mask from the compressed tampered/untampered pair. Moreover, official mask provision rather than frame difference inference removes the philosophical debate over whether a pixel, fully within the tampered region but by chance unchanged by the tampering process, is labelled tampered or not.

3.6 The Future of Tampered Video Datasets and Detection

Many modern techniques of video tampering simply do not fit neatly into the traditional categories of inter- and intra- frame tampering. In particular, there is significant overlap between the recent categories of motion/style transfer and synthetic video generation. Changing the style of a video sequence from semantic segmentation masks to photo-realistic generates a purely synthetic video, but the same techniques can be used

to perform digital puppetry. This means that detection of synthetic video should be viewed as an extension of tampering detection. Given the current trend of using full reference quality measures in the evaluation of retouching, frame interpolation and in video frame prediction, it is clear that one of the current goals is to replicate authentic video. What remains unclear, however, is whether these methods will deviate from authentic content as evaluation methods emerge or even help to launder video tampering evidence in the same way as video compression.

One important new research direction in digital video manipulation is an accepted method of evaluation. Many existing methods rely on only qualitative evaluation and while this is an important first step, adoption of existing video quality techniques, including no reference quality metrics will speed up development. Until then, user studies and public release of manipulated video clips remains the gold standard. In the absence of elegant quality measures, altered video and the associated methods are often publicly released for analysis, and video tampering detectors should look to utilise this provision where possible to create realistic detection methods. To facilitate this, video tamperers should release either sufficient data to simplify the creation of accurate tampering masks or release the masks themselves. Furthermore, video data should be distributed in such a way as to minimise further processing. Video processing, such as compression and retouching can effectively conceal tampering. While detection of such anti-forensics is an important research direction, processing can be applied to video independently after dataset publication, but only if the original dataset is published in such a way as to avoid unnecessary processing. With the increasing application of deep learning methods to tampering detection, any future dataset gatherers must take care to avoid potential pitfalls which cause datasets to reflect their own specific features relating to publishing platform or tool use, rather than those legitimately tied to the tampering technique.

As the variety of video manipulation techniques expands and advances, tampered and synthetic video will become indistinguishable from authentic video to human eyes. Even machine vision techniques are unprepared for the current photo-realistic tampering techniques. To maintain confidence in the authenticity of video content, it is crucial to develop ways to identify and localise video processing and manipulation. Universal video manipulation detection and localisation, irrespective of tampering technique, is essential if tampering detection is to keep pace with tampering methods.

Chapter 4

The Effects of Compression

It is commonly noted that compression is a challenge in video analysis. Techniques which work well on high quality data can suffer from reduced performance or fail completely in the presence of compression. In this chapter we isolate the spatial effects of video compression and provide an experimental framework to examine how these affect performance of CNN classifiers. We confirm that CNN performance is optimised when training data is most similar to the test data, and this is also true in the case of compression. A CNN for classification of heavily compressed video frames will achieve the best performance when it is *trained* on heavily compressed video data. This work demonstrates that conditioning a CNN with compression properties could potentially lead to higher classification accuracy.

The main findings of this chapter were presented at the 2018 International Joint Conference on Neural Networks (IJCNN) as part of the paper "Spatial Effects of Video Compression on Classification in Convolutional Neural Networks" [24].

4.1 Images For Video Analysis

The field of image classification has seen great advances in the state-of-the-art using CNNs. The availability of large datasets such as ImageNet [14] and CIFAR-10 [175] have enhanced the body of research and machines can now surpass humans on some image classification tasks [51]. The natural progression of this research is towards video analysis, and large video datasets already include YouTube-8M [43],

Sport1M [5] and ImageNet's expanding video dataset [59]. There is a repeated tendency, however, to simply transfer all learning from still images straight to video applications without modification. This method of representation may lead to fundamental inaccuracies, which go undetected as large datasets make exhuastive manual checking unfeasible. In the visual object tracking domain, datasets such as [176, 4] are provided only as a sequence of still images. ImageNet [59] provides both video files and extracted JPEG files of the individual, annotated frames. This simplifies algorithm development by precluding pixel extraction from the video file, however any information from the compressed video bitstream is lost, including basic metrics such as frame rate and information about transforms already applied to the pixel data. Moreover, visual comparison of video frames and JPEG files reveals some differences between the two, implying that lossy JPEG compression has been used.

Some prominent fields of video analysis involve feature extraction using CNNs pre-trained on the still images of ImageNet such as the VGG networks of [48] or AlexNet [13]. This includes work in the field of visual object tracking [177, 178, 179, 180, 181], work in the area of video content understanding [182, 183] and in the area of video classification [5, 184]. With such widespread use of compressed video analysis using networks trained on still images, it is worth investigating how video compression affects the features learned in CNNs. Moreover, video tampering detection specifically lists compression and recompression as one of the main challenges in the field [21, 17]. An understanding of *how* compression affects learning in deep neural networks could help to overcome this. Results from this experimental framework shows that performance in CNNs is improved by using the quality of the test data to inform the quality of the training data, rather than the established method of using only the highest quality data for training. This holds true especially in the case of highly compressed video content.

4.2 Compression in Image and Video Analysis

To the best of our knowledge, despite the pervasion of video compression, there has been little investigation into how video compression affects learning in CNNs. The authors of [19] showed how noise resulting from JPEG compression affects image classification in a deep neural network trained on high quality images. Their results suggested that the pre-trained networks were more resilient to compression-related deformations than to Gaussian blur, however, the authors did not consider that the data used to train their network was gathered "in the wild", and most likely already subject to compression. Therefore the results may have been understated. In [185],

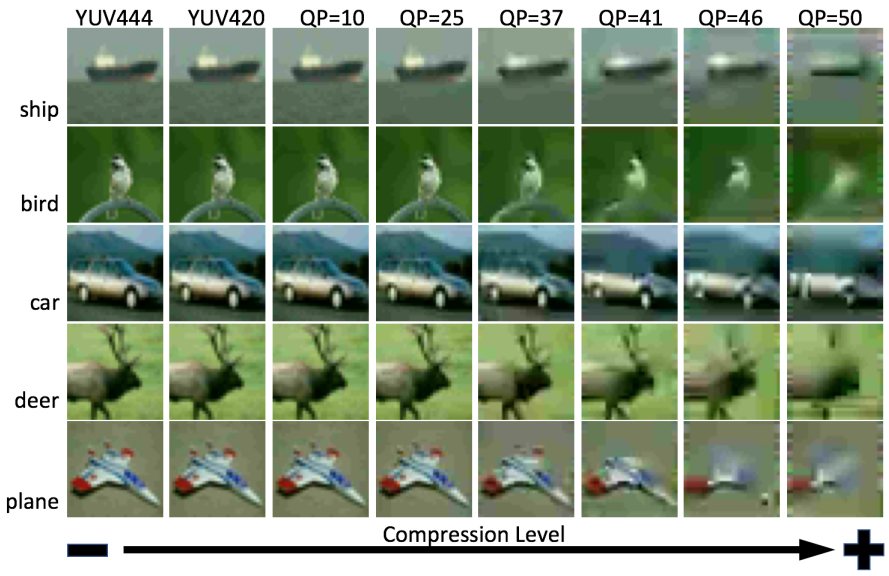


Figure 4.1: The effects of quantisation on images from CIFAR-10. The images in CIFAR-10 are low resolution (32x32 pixels). There is no visual difference between YUV 4:4:4 and YUV 4:2:0 but the number of bytes used to represent the image has been reduced by half

the differences between video and image datasets were also examined in the application of object detection. The authors used a quality metric defined in [186] and rated the quality of their video datasets [186, 59] as lower than that of the image datasets [187, 59]. They attributed this to motion and compression artifacts, but did not investigate compression explicitly. The availability of uncompressed data is a limitation in this field, with uncompressed images limited to very specific datasets such as UCID [166] or Dresden [164] and uncompressed video limited even further to [37] or [1]. Video compression is listed as a challenge in forensic analysis [17], but the mechanism of its effect is seldom examined in isolation.

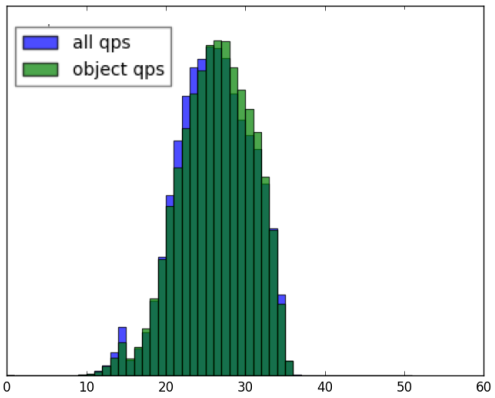


Figure 4.2: The range of QP found in ILSVRC2017 video subset of ImageNet

An overview of the mechanics of compression is given in Section 2.1 and for futher details of H.264/AVC, the reader is referred to [27, 31, 39]. Figure 4.1 gives a visual representation of how H.264/AVC affects the tiny images of CIFAR-10 and Figure 4.3 shows how some images from STL-10 [188] are affected by compression.

All compression can be lossy or lossless. In most cases, video compression is lossy as the first stage in the process usually converts RGB data to YUV 4:2:0, as detailed in Section 2.1.5. This quarters the colour resolution but halves the total number of bytes needed for storage and goes unnoticed by human eyes. This can be observed in Figure 4.1 where the difference between the YUV 4:2:0 and YUV 4:4:4 columns are hardly noticable. JPEG compression can also optionally apply this lossy step, and JPEG subsampling details are included in the JPEG bitstream. In general, CNNs trained on natural images learn distinctive edge-type Gabor filters and colour blobs in the first layer [189]. The use of YUV 4:2:0 or some other chroma sub-sampling method in JPEG training images can explain this: colour blobs are lower resolution than intensity edges, just as the compressed colour component is of lower resolution than the intensity component.

Section 2.1.1 explains how video compression applies quantisation in the frequency domain and Section 2.1.3 details the differences between constant bitrate (CBR) and variable bitrate (VBR) modes of operation. The experiments in this chapter examine both constant quality for reproducibility and constant bitrate for a real-world perspective. For comparison, Figure 4.2 shows a normalised histogram of the spread of QP found in ILSVRC2017 bitstreams for complete frames and the areas within the defined bounding box of the first object in the sequence. There is little difference in QP between the subject of each video and the background, indicating that the encoder used to compress the sequences does not differentiate between the two. The average QP of all frames is 25.52 and the average QP of the first object's bounding boxes is 25.74. The very slightly lower average QP of the background of the videos may be attributed to a larger proportion of skipped macroblocks. Any static background will simply maintain the QP from the last time it was encoded, and higher quality data from intra frames is more likely to persist. More interestingly, the I frames have average QP = 21.93, much lower than the sequence average. The same object will be compressed at different levels of quality in different frames. If a classifier is used to track objects over a sequence of frames, some objects may be missed due to changing compression levels in different frames. The data in Figure 4.2, however, may reflect the ILSVRC2017 dataset itself and the particular encoder used, rather than the real world. Moreover, some of the sequences of ILSVRC2017 exhibit artifacts that are

inconsistent with the QP parameter encoded in the bitstream, suggesting recompression. It is entirely possible to recompress compressed data and thus obliterate any information about the original compression. Furthermore, this may be done multiple times and is largely invisible to the naked eye. If this dataset has been recompressed, then the effects of compression may be understated.

4.3 Examining Video Compression in Isolation

The main motivation of this experiment is to explore how the quality of CNN training data determines performance on test data of the same or different quality. Specifically, we wish to ascertain whether a knowledge of compression parameters could give rise to an improvement in classification accuracy. Because we examine the spatial effects of video compression in isolation and not the temporal effects, it is acceptable to use an image classifier. This also correlates with how CNN object classifiers are commonly used on individual frames of video and it allows for the use of simpler datasets and network architectures. We selected three labelled image datasets (MNIST [190], CIFAR-10 [175], labelled images of STL-10 [188]), from which to synthesise video-frame datasets. This avoids any pre-existing video compression artifacts that may be found in video datasets. Each of the synthesised datasets was then used individually to train a CNN. One CNN was trained using all the synthesised datasets together. Each trained CNN was then tested with all related test sets individually, to see whether features learned using one dataset were immediately transferable to another, closely related dataset.

For the purposes of this experiment, the images in CIFAR-10 were considered uncompressed. CIFAR-10 is based on a subset of Tiny Images [191] where original images were resized to 32x32 pixels. When the image resolution is reduced, so, too, are any spatial compression artifacts. JPEG compression commonly uses an 8x8 block size, so any CIFAR-10 image with original dimensions over 256x256 pixels has blocking artifacts effectively removed. Banding artifacts are also comparatively reduced. To generate a series of uncompressed larger images, the CIFAR-10 dataset was resized to 64x64 (double height and width, Lanczos interpolation [192] for smoothing).

The basic set of experiments was also performed using MNIST. The single binary channel of MNIST was used as the Y-channel in YUV data, with constant values of 128 for the U and V channels to give a greyscale image.

Finally, the labelled images only of STL-10 were used. The images are 96x96x3 and there are 13000 labelled images in the dataset. Prior to dataset synthesis, these were

split into 80% train and 20% test, ensuring an even split of class labels. Like CIFAR-10, this dataset is supplied as RGB pixel values, so there is no way to objectively quantify any previous compression, however compression artifacts can be seen when viewing some of the individual images (Figure 4.3) so these images were not considered uncompressed.

(a) YUV (b) QP=10 (c) QP=25 (d) QP=37 (e) QP=50

Figure 4.3: An example image from STL-10 when compressed at different compressions. STL-10 images are larger than CIFAR-10 (96x96 pixels). Small compression block artifacts visible around the rigging of the ship diminish as the image is compressed but after QP=25, detail is lost.

4.3.1 Dataset Synthesis

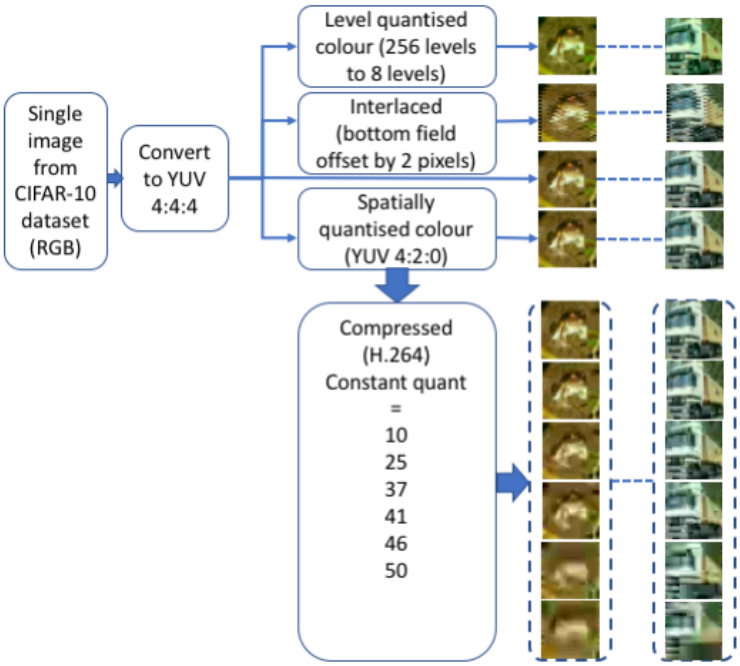


Figure 4.4: Generation of constant quality datasets

Table 4.1: A summary of synthesised datasets

Dataset Name	Description
	The following were applied to CIFAR-10, MNIST and STL-10
YUV 4:4:4	Lossless translation of RGB to YUV colour space
YUV 4:2:0	Lossless intensity (Y); lossy colour (UV) spatially averaged at one quarter resolution
Level UV	Lossless intensity (Y); lossy colour (UV) quantised to 8 uniform levels
Interlaced	Offset alternate horizontal lines by 2 pixels
q*(QP)*_f*(F*	Compressed frame number *(F)* = [0,2,3,6] with Quantisation Parameter *(QP)* = [10, 25, 37, 41, 46, 50];
	The following were applied to CIFAR-10 only
q*(QP)*_f*(F*	Compressed frame number *(F)* = [0,2,3,6] with bitrate *(QP)* $>$ 52

Every training set and every test set (Table 4.1) was synthesised using the same split of original images. That is, for each of the original datasets (CIFAR-10, CIFAR-10-doubled, STL-10, MNIST), the train/test split was decided prior to any synthesis. The official splits were used for MNIST and CIFAR-10. This resulted in images across related datasets that were visually similar (Figure 4.1) but with no data leakage between test and train sets.

Figure 4.4 gives an overview of how different video-frame datasets were synthesised from CIFAR-10, STL-10 and MNIST. First, RGB images were transformed into YUV 4:4:4 (in the case of MNIST, the intensity values simply formed the Y channel with the U and V channels set to grey/128). Then the UV chroma component was spatially averaged to translate to YUV 4:2:0. The YUV 4:2:0 frames were then used to generate the compressed frames . As a side experiment, we also quantised the UV component to 8 fixed colour levels. This has little visual effect but helped to gauge the impact of colour data on CNNs.

Interlaced frames were produced by offsetting every alternate row of YUV 4:4:4 by 2 pixels. Interlacing (Section 2.1.6) is a historical technique in broadcast video where the top field (odd rows) and the bottom field (even rows) are captured individually. Our simulation of interlacing mimics the comb effects of a slow camera pan over a static scene. In the real world, the extent of these effects depend on object motion relative to the camera, and different objects in the same frame exhibit different degrees of combing.

The YUV 4:2:0 dataset was used to synthesise compressed video frame datasets (Figure 4.4). Still images were used to make a simple 7-frame video (Figure 4.5).

Figure 4.5: The 7 frames of a short, synthesised video. The invariant image is overlaid onto a moving background of upwards scrolling horizontal lines. The background border adds 8 pixels on each side so that of the 9 macroblocks comprising the image, 8 contain moving data. This forces the compression codec to use predicted blocks with motion compensation, rather than skipped blocks as it would with a repeated image.

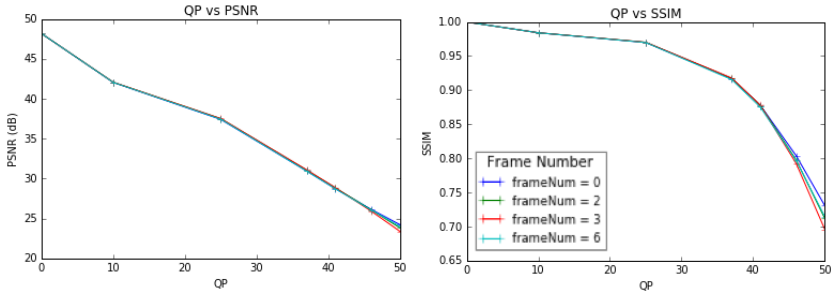


Figure 4.6: The effects of quantisation on image quality metrics

This sequence was compressed using constant QP or bitrate, a Group Of Pictures (GOP) structure of IBBPBBP and one-pass encoding. Bitrates and quantisation parameters were manually selected to allow a range of quality in the video sequences. The extracted frames were of the frame types: **I**ntra frame (0); **B**i-directional frame (2); **P**redicted frame (3,6).

The original images of CIFAR-10 are 32x32 pixels and the moving border adds 8 pixels on to top, bottom, left and right. The resulting uncompressed video is 48x48 pixels, or 3x3 macroblocks in video compression terms. It is important to note that any static parts of a video frame will be largely encoded as skipped macroblocks unless there is a change in quantisation due to rate control. The moving border is half a macroblock wide so out of 9 macroblocks making up each frame, only the centre macroblock will remain unchanged.

Constant Quality

In the first experiment, all rate control parameters and psychovisual enhancements were disabled and constant QP across all frames was used. The QP values used were 10, 25, 37, 41, 46, 50. These values were selected to give a broad range of quality based on PSNR and SSIM. Figure 4.1 shows the visual effect of this on some images from CIFAR-10. Figure 4.6 shows how constant QP maps to both PSNR and SSIM

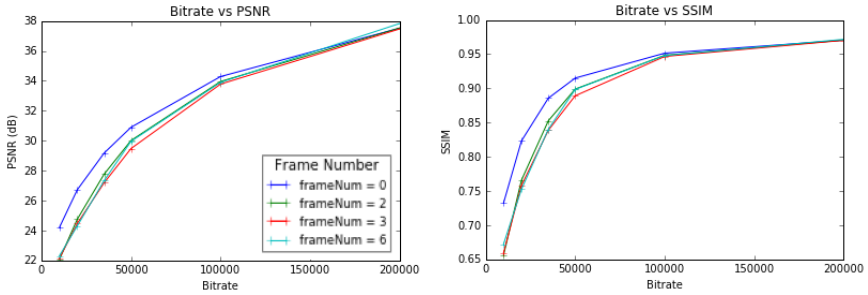


Figure 4.7: The effects of bitrate on image quality metrics

with regards to the compressed video frame datasets. PSNR is inversely proportional to QP as designed, whereas SSIM has a much sharper drop off after QP 37 which tallies more closely with the visual representation. The graphs in Figure 4.6 also show that with a constant QP across all frame types, there is very little difference in the quality of the different frame types of each sequence. The constant quality experiment was performed using CIFAR-10, MNIST and STL-10 datasets.

Constant Bitrate

In the second experiment, the rate control parameters and psycho-visual quality settings of x264 were enabled on only CIFAR-10. To keep dataset synthesis computational requirements low, one-pass encoding was used and only the 7 frames of the synthesised video sequence were encoded. Bitrates were manually selected to allow a good range of quality in the video sequences. Rate control algorithms typically take some time to initialise fully, so with a very short 7-frame sequence, the requested bitrate may not be indicative of the actual achieved bitrate. The graphs in Figure 4.7 show how bitrate affects the quality metrics. It is common for encoders operating at low bitrates to encode the key frames of a sequence with a slightly higher quality, particularly in the case of the first frame in the sequence and particularly when bitrate constraints are tight. This increases the similarity between the current frame and the reference frame used for prediction and therefore allows more efficient compression. This explains why the key frame 0 quality curves are slightly higher than those of the predicted frames. It is worth noting that the shape of the PSNR graph in Figure 4.7 does not relate directly to that in Figure 4.6: Figure 4.7 is more curved, Figure 4.6 is more linear. This demonstrates that the image quality is reliant not just on QP but also the quality of the data available for prediction.

The constant bitrate experiment was performed using only CIFAR-10.

4.3.2 Double Image Dimensions

It can be said that using tiny images and compression a frame of 48x48 pixels overstates the effect of compression because any compression artifacts are comparatively large compared to the data. To investigate this, the CIFAR-10 dataset was resized to 64x64 (doubling height and width, using Lanczos interpolation from Python's PIL library) and the experiments repeated, using appropriate bitrates for compression and, again, adding an 8-pixel moving border to simulate motion. The network architecture was the same as used in the previous experiment. The network was trained using random 48x48 crops (as in the original bitrate experiment, the network was trained using random 24x24 crops).

4.3.3 Network Architecture

The network used in the experiments is conv5x5-64, pool3x3, conv5x5-64, pool3x3, fc-384, fc-192, softmax. ReLU was used. The learning rate was fixed at 0.1. This network, similar to those in [46], is known to achieve a precision of around 84% on RGB CIFAR-10 after 30k iterations and approximately 86% after 60k iterations. The purpose of these experiments is not to directly enhance state-of-the-art but to investigate the effects of video compression on learning in a controlled way.

Frames derived from CIFAR-10 were normalised by subtracting the average pixel value from all pixels. For data augmentation, the images were randomly cropped from 32x32 to 28x28 and randomly horizontally flipped during training. The test set was centrally cropped and normalised. MNIST frames were normalised but had no further augmentation applied. STL-10 derived frames were normalised and flipped only.

An alternative, deeper network was also used for STL-10. The modified network architecture added an additional conv5-64 layer giving an overall architecture of: conv5x5-64, pool3x3, conv5x5-64, pool3x3, conv5x5-64, pool3x3, fc-384, fc-192, softmax.

It was found that switching from RGB24 colour space to YUV 4:4:4 had no overall impact on precision, as intuitively expected.

4.3.4 Training and Testing

A model was trained on every dataset in the series and each model was tested with every related dataset (Table 4.2) and the mean Average Precision (mAP) recorded.

Table 4.2: Related datasets

Source dataset	Synthesised (related) datasets
CIFAR-10	YUV 4:4:4, YUV 4:2:0, Level UV, Interlaced, q*(QP)*_f*(F)*; *(QP)* = [10, 25, 37, 41, 46, 50]; *(F)* = [0,2,3,6]
CIFAR-10	YUV 4:4:4, YUV 4:2:0, Level UV, Interlaced, q*(QP)*_f*(F)*; QP $>$ 52; *(F)* = [0,2,3,6]
CIFAR-10-double	YUV 4:4:4, YUV 4:2:0, Level UV, Interlaced, q*(QP)*_f*(F)*; *(QP)* = [10, 25, 37, 41, 46, 50]; *(F)* = [0,2,3,6]
MNIST	YUV 4:4:4, YUV 4:2:0, Level UV, Interlaced, q*(QP)*_f*(F)*; *(QP)* = [10, 25, 37, 41, 46, 50]; *(F)* = [0,2,3,6]
STL-10	YUV 4:4:4, YUV 4:2:0, Level UV, Interlaced, q*(QP)*_f*(F)*; *(QP)* = [10, 25, 37, 41, 46, 50]; *(F)* = [0,2,3,6]

*Table 4.3: Mean Average Precision for networks cross evaluation on CIFAR-10 intra frame. Best trained network for this test underlined; Best test result for this trained network in **bold**.*

	Trained On (QP)							
Tested With (QP)	All	YUV	10	25	37	41	46	50
YUV	73.6	**84.0**	83.4	**82.6**	78.8	75.0	67.7	60.9
10	73.7	83.6	**83.5**	**82.6**	**79.3**	75.1	68.1	61.0
25	73.5	82.1	82.5	82.2	79.0	**75.4**	68.0	61.0
37	70.6	73.1	73.5	74.6	76.9	74.4	**68.8**	61.0
41	67.6	63.6	63.5	65.7	71.4	72.3	67.9	**61.6**
46	60.1	48.4	48.0	49.7	58.2	62.3	64.2	59.9
50	53.3	36.1	36.7	37.7	44.4	50.3	57.0	57.9

To examine the effect of **pre-training**, a CNN was initialised with all the weights and biases learned on one synthesised dataset and allowed to further train on another. Specifically:

- pre-trained on YUV 4:4:4, further trained on uncompressed YUV 4:4:4
- pre-trained on YUV 4:4:4, further trained on highly compressed QP 50
- pre-trained on highly compressed QP 50, further trained on YUV 4:4:4
- pre-trained on highly compressed QP 50, further trained on QP 50

The weights learnt in the first layer were then examined to find out if all weights responded equally to further training.

4.4 The Influence of Compression

One of the main findings is that CNNs respond to compressed video in an intuitive way. In general, more compressed video leads to lower classification precision (Figure 4.9, Table 4.3). The effects of heavy compression distort objects enough to render them unrecognisable by human eyes (Figure 4.1), and the results (Figure 4.9) show that this is broadly matched in machine vision. The maximum precision for a given test set is generally achieved by a network trained with data most similar to that of the test set. A network trained with a variety of different compression levels becomes a “jack

(a) YUV (b) QP=10 (c) QP=37 (d) QP=50 (e) Double (f) interlaced

Figure 4.8: The first layer filters, constant QP training on CIFAR-10 (re-ordered according to variance in each filter)

of all trades, master of none" (Table 4.3).

Features learned on one compression level transfer well to less compressed video: a network trained on video frames compressed with a particular QP achieves at least the same precision on frames compressed with a lower QP. Similarly, a network trained on video frames compressed at a given bitrate performs equally well or better when tested with frames compressed at a higher bitrate. Conversely, although a network trained on high quality frames achieves better precision when tested with frames of high quality, it experiences greater drop off in precision when faced with lower quality data.

4.4.1 Colour Spaces

In datasets with no video compression (YUV 4:4:4, YUV 4:2:0, level UV and interlaced), features trained on **interlaced** frames do not transfer well to non-interlaced frames (Table 4.4). Moreover, networks trained on non-interlaced data did not achieve good precision on interlaced test data. Features learned on interlaced data were more transferable to non-interlaced data than vice versa. Visualisation of the filters learned in the first layer of the CNN (Figure 4.8) shows some combing effects. It can be theorised that features from CNNs trained only on still image data or photographs which do not contain interlaced data will perform less well on interlaced videos. This effect diminished with a larger frame size (Table 4.4), so smaller features may be more affected by interlacing, but further research is needed to confirm this. It is common to resize video and images for neural networks [13, 3], and this reduces combing effects. A de-interlacing filter may be applied but this will cause some blurring. Although this is not examined here, we hypothesise that de-interlacing artifacts will reduce CNN performance.

Decreased **colour information**, either by quantisation to fixed levels or by reduction of colour resolution, does not have a large impact on maximum achievable precision

Table 4.4: Mean Average Precision for networks trained/tested on different uncompressed colour spaces generated from CIFAR-10 [single image dim; double image dim]

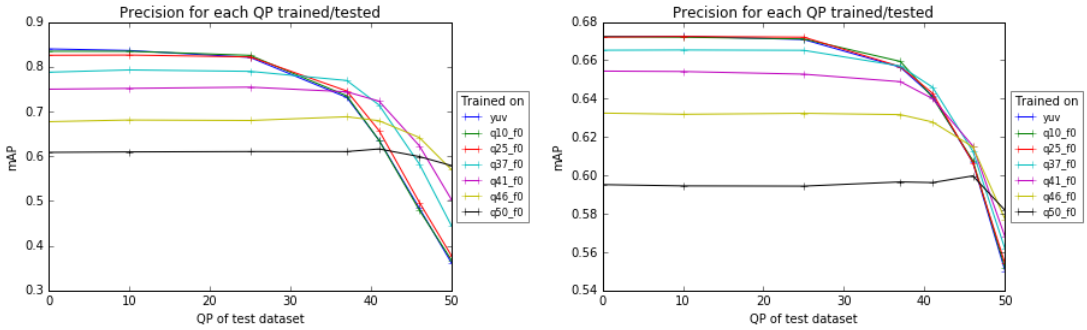
	Trained On			
Tested With	YUV 4:4:4	Level UV	YUV 4:2:0	Interlaced
YUV 4:4:4	84.0; 82.0	83.4; 81.1	83.4; 81.8	59.3; 53.8
Level UV	83.3; 80.1	83.6; 81.2	82.8; 80.2	57.9; 53.3
YUV 4:2:0	83.7; 81.8	83.2; 80.7	83.4; 81.5	58.2; 53.0
Interlaced	37.3; 78.2	39.8; 76.5	33.0; 78.2	82.0; 81.6

(Table 4.4). A drop in colour data in the training set or the test set leads to a drop in mean average precision, however reducing colour resolution to one quarter of its original value (as in YUV 4:4:4 to YUV 4:2:0 conversion) does not significantly impact classification (mAP difference of 0.6%). This may be partly attributed to the shape of the learned filters: it is possible that the most discriminative filters, like the human visual system, weight luma (intensity) more strongly than chroma (colour).

4.4.2 Constant Quality

The graph in Figure 4.9 shows the how same network architecture trained on differently quantised datasets performs when given test data with different QP. Table 4.3 gives more detailed results for CIFAR-10. The maximum achievable mAP generally reduces as quantisation of the training data increases. Interestingly, the maximum mAP for each network does not necessarily coincide with testing on its related dataset (Table 4.3). The network trained on data with QP=50, for example, gives its lowest mAP when tested with QP=50 data. In general, slightly higher precision is achieved when a network is tested with data of a lower QP than than of its training set. Conversely, test data with a specific QP achieves the best precision when passed through a network trained on a similar QP.

The plots of the networks trained on STL-10 are very close together for uncompressed, QP=10 and QP=25 (Figure 4.9). Using the deeper network architecture for STL-10, it was possible to improve accuracy slightly. Interestingly, deeper networks trained on QP=10 and QP=25 outperformed those trained on uncompressed data. The results are shown in Figure 4.10. This can be partly explained by visible compression artifacts in the original image (Figure 4.3). Some original STL-10 images show blocking artifacts that are visibly reduced by compressing with QP=10, and further reduced with QP=25. This is a serendipitous effect of recompression. In compressed image data, compression artifacts might occupy specific frequencies that have otherwise been lost due to quantisation. Recompressing the compressed image at a similar level might



(a) CIFAR-10 (b) STL-10

Figure 4.9: Precision for each QP trained/tested. Features learned on a given QP transfer well to test data encoded with the same QP or lower, best viewed in colour

disproportionately reduce the compression artifacts while having little effect on the image data itself. At QP=37 and above, introduced compression artifacts come into play and the maximum mAP is consequently reduced.

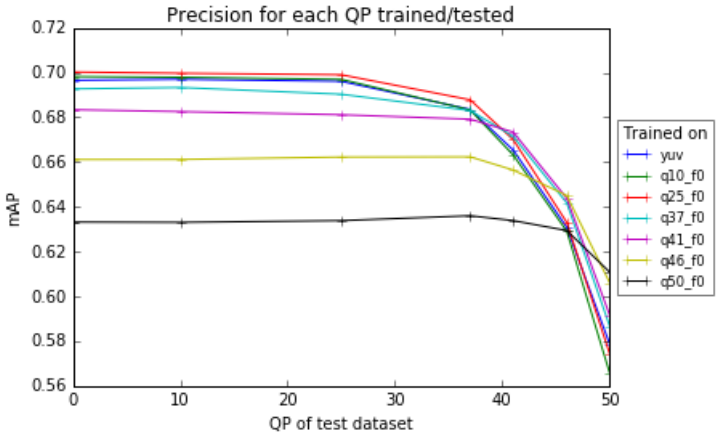


Figure 4.10: Results for a deeper network on STL-10 which achieves the best result when trained on QP=25.

The first layer filters of a neural network are known to give an indication of the low level features learned in the network. An examination of these for the network trained on CIFAR-10 data (Figure 4.8) shows that networks trained on higher QP data rely more heavily on colour-based filters rather than intensity filters. Intensity filters learned on high QP data are far less distinct than their low QP counterparts. This corresponds with the idea that the effects of quantisation in the frequency domain manifest visually as blurring. The colour filters themselves represent colour transitions rather than discrete colour blobs, although this may be explained by the size of the convolution kernel: in a 5x5 kernel, there is more room for complex patters to emerge.

It was found that MNIST data was very robust against video compression. Video compression applied to MNIST gave a range of SSIM from 0.74 and PSNR from 21.0 dB. Application of constant quality compression to CIFAR-10, yielded minimum SSIM = 0.36 and PSNR = 16.9 dB. For STL-10: minimum SSIM = 0.42 and PSNR = 19.41 dB. While PSNR gives an indication of how the compressed signal differs from the original, SSIM gives a better indication of *visual* difference. The networks trained on datasets synthesised from MNIST showed little change in performance when tested with data of a different compression level. The pattern of increasing QP leading to decreasing mAP was still present, however the mAP ranged from 97.0% to 99.2% for non-interlaced data. and 95.0% to 99.3% for interlaced data.

The range of SSIM shows how compression affects the data. The smaller range of quality metrics for compressed datasets synthesised from STL-10 shows that CIFAR-10 is more adversely affected by compression and this relates to the range of mAP (Figure 4.9). Datasets derived from STL-10 have a lower range of SSIM *and* mAP than those derived from CIFAR-10. This can be attributed to the way that the images were turned into videos. In the larger STL-10 frames, the moving border occupies a proportionately smaller area than for the CIFAR-10 images. The central portion of the frame unaffected by the border is proportionately larger in the STL-10 images. Given that the border pixels are arbitrary rather than related to the image content, the encoder will be able to make better predictions using data from the image itself, and when the residuals themselves are heavily quantised by a high QP, good predictions help improve quality a lot. In Figure 4.1, the effects of the horizontally striped borders can be seen in high QP images. In Figure 4.3, these effects are proportionately smaller, although they can still be seen on the left and right borders of Figure 4.3e.

4.4.3 Frame Type

When using constant QP, frame-type (I, B, P) has little effect on CNN classification, with average variation of 0.6% mAP, and slightly better performance for networks trained on frame 0. This ties in with the quality metrics shown in Figure 4.6 where it can be seen that, with constant QP, there is very little difference between the frame types. For constant bitrate, this effect was more marked, with an average range of 1% mAP and networks trained on frame 0 exhibiting higher performance. Similarly, testing with I-frames also achieves higher performance than P- or B-frame test batches. This can be attributed to rate control mechanisms in x264 which allocated more bits to I-frames in our sequences, and thus a lower average QP as illustrated in Figure 4.7.

4.4.4 Constant Bitrate

The graph of networks trained on datasets of different target bitrates (Figure 4.11) shows similar results to those for constant QP: highly compressed data yields lower achievable mAP, but networks trained solely on uncompressed or high bitrate data perform poorly when classifying low bitrate frames.

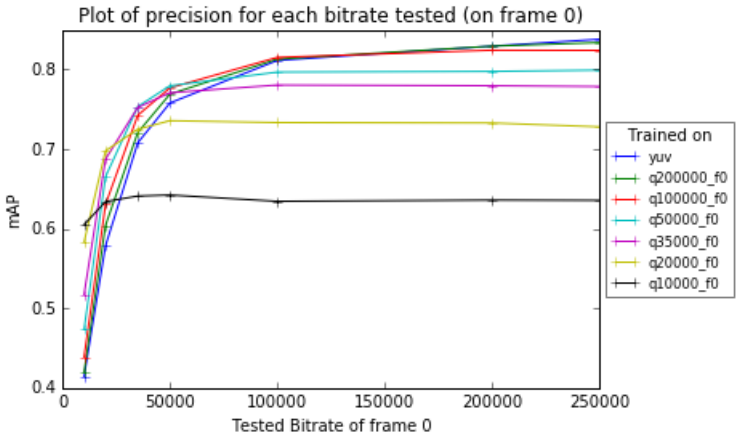


Figure 4.11: Precision for each bitrate trained/tested. Features learned on a given bitrate transfer well to test data encoded with the same bitrate or higher

The Figure 4.12 shows a plot of SSIM vs precision. For each network trained on data compressed at a specified bitrate, test data at a different compression rate is compared with the test data of the network's own compression rate to get the SSIM measure which is plotted against the mAP for that test data. The datapoints with SSIM=1 are the networks trained and tested on the same level of compression. It could be theorised that a network trained on a given level of compression would achieve its best precision when faced with test data most similar to its training data. This graph shows that this is *not* the case. Networks trained on highly compressed, low bitrate data still perform better on data with a higher bitrate. The worst performance by the network trained on the lowest quality data is achieved when testing with the lowest quality data. At the other end of the scale, however, networks trained on the highest quality data suffer the most degradation in performance when faced with lower quality data and the worst mAP overall is achieved by testing the most compressed data on the network trained on uncompressed data.

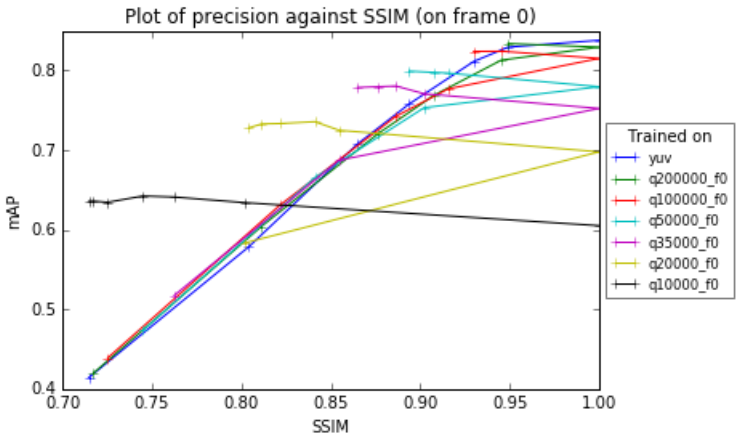


Figure 4.12: Precision for each SSIM trained/tested.

4.4.5 Double Image Dimensions

For double image dimensions, results were broadly similar to those found above: networks trained on uncompressed data achieve the highest mAP of 82% but this drops off most quickly when tested with lower quality data. Networks trained on lower quality data achieved lower maximum mAP and produced equal or better mAP when tested with higher quality data. Most interestingly, Table 4.4 shows that the drop in performance for interlaced data was much lower in this experiment. This can be attributed to the network learning spatially larger discriminative features (Figure 4.8) and also because the interlaced offset was maintained as 2 pixels so the combing effect was proportionately smaller on larger images. A further experiment was performed using the deeper network architecture with an extra conv5-64 layer. Results showed improved performance (7% absolute mAP) of the interlaced-data-trained network when tested with non-interlaced data. This suggests that deeper layers may effect some form of deinterlacing but further research is needed.

Another intuitive result is in the visualisation of the first layer filters (Figure 4.8): doubling the image dimensions of the dataset without changing the kernel size of the first layer simply doubles the learnt feature size. This is significant because it helps explain the common appearance of first layer filters trained on natural images. If training images utilise YUV 4:2:0 colour space, then chroma resolution is one quarter of luma resolution. Therefore, colour features learned in CNNs are also lower resolution than their intensity (edge-type) counterparts. This implies that the use of YUV 4:2:0 is widespread in JPEG data used to train many modern networks. Hypothetically, the first layer features of a network trained exclusively on YUV 4:4:4 data would exhibit colour-edge features rather than colour blobs.

4.4.6 Pretraining

The results for pre-training (Figure 4.13) show a network can be fine tuned towards different levels of compression but this reduces precision for the original data. In the case of a network pre-trained on uncompressed data (Figure 4.13, top row), the differences in the first layer filters are very slight but the differences in mAP are marked. This suggests that much of the fine tuning is achieved in higher layers of the network. For the network pre-trained on uncompressed data and further trained on highly compressed data (Figure 4.13, top right), the edge-type filters in the first layer do not visually blur. It can be hypothesised that the increase in mAP from 36.1% to 58.3% comes from the equivalent of feature blurring in deeper layers. Sharp features in the first layer are required for mAP on uncompressed data.

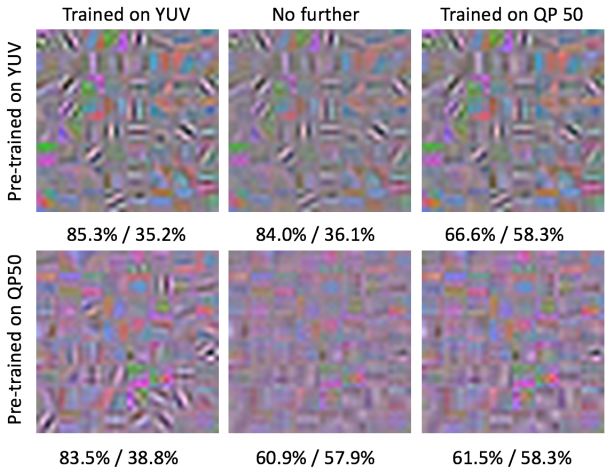


Figure 4.13: The filters learned in the first layer with pre-training: the central column shows the original trained network with no further training, the left column has been further trained with uncompressed data, the right column has been further trained with the most compressed data [mAP on uncompressed / QP 50 data]

Pre-training a network with QP = 50 and further training with uncompressed data (Figure 4.13, bottom left), allows sharp edge-type filters to emerge in the first layer and the mAP for uncompressed data improves, at the expense of highly compressed data. Further training with compressed data might allow deeper layers of the network to generalise around these sharp features and improve compressed test precision. The mAP improves on both compressed and uncompressed test sets when the network is further trained on highly compressed data (Figure 4.13, bottom right). This improvement is small but follows the trend shown in Figure 4.9 where features learned on compressed data transfer successfully to less compressed datasets.

4.4.7 Limitations

This work is limited by the availability of uncompressed data. The data used in the experiments in this chapter comes from rescaled images, the original compression of which cannot be truly verified. The ideal dataset would be uncompressed, clean, short, independent, labelled video sequences, but such data is in short supply. It is reasonably assumed that the dimensions of the images of CIFAR-10 have been reduced sufficiently so as to be considered uncompressed.

We synthesised our own video datasets from still images for these experiments. Intra frame compression can be validly performed on single frames but in compressing artificially constructed video sequences, video encoders may make unnaturally constrained compression decisions, particularly when compressing predicted frames. Thus the effects of *frame type* may be understated in these experiments.

4.5 Conclusions

With high prevalence of compressed video and increasing use of CNNs for video analysis, we have provided a timely evaluation of the effects of video compression on classification by CNNs. We found that the highest precision on a given test set occurs when the network is trained on a similarly compressed dataset. Features learned in networks trained on highly compressed frames are equally valid when testing less compressed data, but the precision of the network is generally lower than that of one trained on uncompressed frames. A network trained on images of a specific quality disproportionately misclassifies objects of a worse quality but mostly maintains performance when classifying data of a similar or better quality. Networks trained on highly quantised video frames learn more colour features in their first layer than networks trained on high quality frames which learn more edge-type features. First layer colour features perform better than edge features for classifying low quality frames, but it is possible for higher layers to compensate for this when a network has been pretrained on uncompressed data. A network trained on a variety of levels of compression achieves lower precision than specialised networks, so using very different levels of compression as a form of data augmentation will not improve performance on all data.

We have also explained how the use of compressed YUV 4:2:0 shapes the first layer filters of CNNs trained on natural images. Lower colour resolution in training data means the filters diverge into higher resolution edge-type filters and lower resolution

colour blobs. Higher resolution colour features may emerge from CNNs trained on uncompressed images which use YUV 4:4:4 colour space. The first layer filters also show that the effects of compression are present in the pixel level of video and CNNs cannot necessarily compensate in deeper layers. If compression artifacts which are invisible to the naked eye influence *object* classification accuracy, then they will certainly disrupt tampering detectors which are trained on high quality tampered data.

The work in this chapter suggests that information about a compressed video bitstream, such as quantisation, can inform models for video analysis and improve performance. Unfortunately, videos can be recompressed, obliterating information about past compressions from the bitstream. The following chapters show how compression parameters can be deduced directly from pixels, thus by-passing any challenges associated with the recompressed bitstream.

Chapter 5

Learning Directly from Pixels

In Chapter 4, the findings show that knowledge of compression parameters may improve accuracy in CNNs for video analysis. In Chapter 3, we discuss how compression and recompression makes tampering detection more challenging. Recompression can be ascertained whenever there is a mismatch between compression parameters encoded in the syntax elements of the compressed bitstream and those derived from the pixels themselves. However, deriving compression parameters directly and solely from the pixels is not trivial. In this chapter, we propose a new method to estimate the H.264/AVC quantisation parameter (QP) in frame patches from raw pixels using Convolutional Neural Networks (CNN) and class composition. We show that, subject to some limitations, the level of quantisation can be estimated directly from pixels themselves using CNNs trained on synthesised datasets. Development of these features opens new, interesting research directions in the domain of video tampering/forgery detection.

The main findings of this chapter were presented at the 19th International Conference on Engineering Applications of Neural Networks (EANN) [25].

5.1 Why Are Compression Features Useful?

Today is the age of fake news and falsified video, and the detection of forged video has become an important area of research. Machine learning techniques can be used to alter video content by, for example, changing faces [64] or weather conditions [65], yet detection of such tampering remains an open field. Detection methods can be active or passive [21, 87], but, since many existing videos are unprepared for active

tampering detection, passive detection methods are more versatile. Passive tampering detection have traditionally been categorised as recompression, region tampering and inter-frame forgery [21], and the latest digital video manipulation methods [60] may cause a demand for a new category of synthetic video detection. Regardless of the editing method, however, any tampering at the pixel level of a compressed video requires recompression of the video bitstream [70, 193], and may leave distinct traces on the pixels. In order to find those traces, we must look at estimating compression parameters directly from pixels themselves.

Video compression is prevalent in digital society. The vast majority of online video has been compressed using lossy formats such as H.264/AVC [27] or MPEG-2 [28]. These formats have been designed with the human visual system in mind and the effects of compression remain largely unnoticed by human eyes. It has been shown that compression does impact classification performance of convolutional neural network (CNN) classifiers [19, 24] and pre-existing compression in original source images may even have caused these effects to be understated.

An intuitive indication of recompression is where the Quantisation Parameter (QP) encoded within the bitstream fails to match the value estimated from the pixels. This is most obvious to human eyes when bitrate and syntax elements of the bitstream imply high quality video data but pixel content exhibits visible compression artifacts such as blockiness. Accurate QP estimation from pixels would allow recompression detection and may aid tampering detection and localisation. The human visual system cannot distinguish between close QP levels, however, and objective methods of measuring QP from pixels are required. In addition, an ideal QP estimator would operate accurately over small patches to enable localisation of tampered regions because the most convincing lie is based on truth, and authentic pixels form the majority in many tampered videos. Tampering localisation is an advancing area of research [77, 87]. For singly compressed frames, estimated QP can be verified by encoded bitstream syntax elements. In video which has been compressed more than once, recompression can be established when there are mismatches between estimated QP and syntax elements. Moreover, a QP estimater can be used to detect any differences in QP patterns between tampered and authenthic regions.

5.2 A Multivariable Problem

The human visual system is adequate to detect some compression effects and can quantify “no reference” image and video quality [194, 195]. The source of video

compression visual effects can be found by examining transformations and mechanisms used in compression standards. As discussed in Section 2.1, a video sequence comprises key frames, which provide access points into the sequence, and predicted frames which rely on data from previously encoded frames. Macroblocks of pixels or pixel residuals are transformed using Discrete Cosine Transform (DCT), quantised with varying Quantisation Parameters (QP) and then variable length encoded into a bitstream. The choice of QP varies according to the rate control algorithm of the individual encoder and the bitrate requirements. A lower bitrate requirement will result in increased QP. An increase in QP often manifests visually as an increased "blockiness"; that is, discrete regions of macroblocks consisting single or few frequency coefficients which makes the borders of macroblocks more noticable. Most often, low frequencies have higher signal amplitudes, so sharp edges persist while textures are reduced. In key frames, macroblock edges align uniformly within the frame, and this can further increase blockiness as the aligned macroblocks form a perceptible grid. Compression artifacts are not restricted to artificial block edges, however, and can also manifest as a lack of specific frequency detail or as banding in areas of smooth colour/intensity transition. See Section 2.1.6 for more details on compression artifacts.

The problem is that compression artifacts and the visible effects of compression depend on many variables. Video content, encoder choices, bitrate requirements can all contribute to how a video is compressed and also how compression artifacts present themselves. Identifying the level of compression directly from the pixels themselves is a complex task and one best handled by deep neural networks.

5.3 Existing Features of Compression and Recompression

Traditional methods of recompression detection rely on the identification of patterns in frequency domain bitstream syntax elements. The authors of [196] rely on Benford's distribution of DCT coefficients and support vector machines to detect double compression of intra frames. In [197] multiple compression is detected in H.264/AVC encoded videos but the compression modes are heavily restricted and the methods do not differentiate between QP that are less than two steps apart. The authors of [198] examine motion features in the bitstream, identifying low and high motion regions and examining differences in the encoded residual in these areas.

As noted in [21], many inter-frame tampering detection methods struggle to detect tampering that aligns with key frames. Methods fail when a complete Group of Pictures (GOP) from one key frame to the next is deleted, added or temporally moved. It can

be deduced from this that these techniques ultimately rely on a detected mismatch between key frames identified using features from the pixels and those derived from the bitstream syntax elements. It is clear from this that there are some differences between intra and predicted frames in video compression.

As part of an investigation into using deep neural networks to determine image quality, Bosse et al [194] developed a method to estimate QP of HEVC frames directly from pixels. Accurate results were achieved for average QP estimation over a complete frame using a patch-wise technique and dataset synthesised from UCID [166]. The method was applied to intra (key) frames only. QP estimation was framed as a regression problem and the dataset used to train the network contained labelled patches compressed with all possible QPs. Although the averaged QP prediction for a complete frame was accurate, a heatmap showing individual patch contributions displayed great variation between patches. If QP estimation is to be successful as a region-tampering detector, it should be as accurate as possible over small regions. Moreover, a QP estimator for video should also handle *predicted* frames, if possible.

This work examines QP estimation in the context of patches taken from H.264/AVC video sequences. H.264/AVC is currently one of the most popular video compression standards and is used on YouTube, broadcast video and public datasets. A CNN is trained to classify frame patches from a video sequence using their quantisation parameters as labels. Unlike [194], we also investigate predicted frames in a video sequence.

5.4 Estimating Quantisation Parameters from Pixel Patches

5.4.1 Synthesising Datasets

When examining the effects of compression, is vital to start with unprocessed data. Standard YUV 4:2:0 sequences from xiph.org are commonly used for video compression quality analysis [37]. Strictly speaking, YUV 4:2:0 is a compressed format due to reduced resolution of the colour channels but it is widely used in video compression. The sequences from xiph.org come in various dimensions and cover a wide variety of subjects from studio-shot sequences to outdoor scenes. All sequences are single camera, continuous scenes. Camera motion varies between sequences but frames from a single sequence will be correlated.

A large amount of data is required to train a neural network and uncorrelated data

will produce a more generalised network. It is possible to use still image data as single frame sequences when focussing on spatial compression artifacts and excluding temporal compression. For this purpose, the images of UCID [166] were used. UCID consists of uncompressed images which are either 512x384 pixels or 384x512 pixels and cover a wide variety of subject matter. All are natural scenes and taken with the same camera. Of the original reported 1338 images in the dataset, only 882 were available for download[1]. Using a dataset of single images is not ideal since predicted frames cannot be examined. However it allows for a greater variety of pixel combinations in a smaller dataset because individual images are uncorrelated. Each image from UCID was regarded as a single frame video sequence and encoded accordingly as an intra frame.

Table 5.1: A summary of original datasets, taken from publicly available sources

Name	Source	Length	Dimensions	Key frames
CIFvid	xiph.org	18 videos	352x288	1/250
CIFintra	xiph.org	18 videos	352x288	all
AllVid	xiph.org	44 videos	176x144 to 1920x1080	1/250
AllIntra	xiph.org	44 videos	176x144 to 1920x1080	all
UCID	UCID [166]	882 single frames	512x384 or 384x512 pixels	all

Table 5.1 gives a summary of the original datasets. Each video sequence was compressed using the open source H.264/AVC encoder x264 and one of a range of constant QP levels using variable bitrate mode. H.264/AVC was chosen because it is currently the most widely used video compression standard. Constant quantisation parameters were selected with an even distribution: QP=[0, 7, 14, 21, 28, 35, 42, 49]. Constant bitrate rate control, psychovisual options and deblocking filter were turned off. For datasets containing predicted frames, the key frame interval was 250. Patches were then extracted from the decoded YUV 4:2:0 sequences. Patches were converted from YUV 4:2:0 to YUV 4:4:4 by upsampling the data in the colour channels. Table 5.2 summarises synthesised datasets. A large temporal stride was used to limit correlation between patches. Consecutive frames are similar to each other and training a neural network with a correlated dataset will cause overfitting. Each patch was labelled with its quantisation parameter. All datasets were prepared in advance of network training and the original video sequences were split into train and test sets prior to compression and patch sampling to prevent data leakage[2].

Two different patch sizes were selected to investigate which aspects of compression

[1] UCID images from http://jasoncantarella.com/downloads/ucid.v2.tar.gz

[2] Training sequences: akiyo, bridge-close, bridge-far, carphone, claire, coastguard, foreman, hall, highway, mobile, mother-daughter, paris, silent, stefan, waterfall, old_town_cross, crowd_run, ducks_take_off, in_to_tree, mobcal, old_town_cross, parkrun, shields. Test sequences: bus, flower, news, tempete

Table 5.2: A summary of synthesised datasets

Name	Source	Patch Size	Spatial Stride	Temp. Stride	Train Patches	Test Patches
AllVid_80	AllVid	80	80(train); 40(test)	40	156592	8400
AllIntra_80	AllVid	80	80(train); 40(test)	40	156592	8400
UCID_80	UCID	80	80	1	131904	53480
CIFvid_80	CIFvid	80	48	30	79920	7920
AllVid_32	AllVid	32	80(train); 40(test)	40	191776	13872
AllIntra_32	AllVid	32	80(train); 40(test)	40	191776	13872
UCID_32	UCID	32	80	1	183320	26320
UCID_32_large	UCID	32	32	1	974528	140512
CIFvid_32	CIFvid	32	32	60	118976	12672
CIFintra_32	CIFvid	32	32	60	118976	12672

were important to CNNs. Block edge artifacts in intra frames will present themselves at macroblock (and subblock) boundaries. Therefore, any patch size larger than 16x16 will capture block edge artifacts. Following [194], a small patch size of 32x32 was selected. A larger patch size of 80x80 pixels was also used. When aligned with the macroblock grid, 80x80 pixels covers 5x5 complete macroblocks. A larger patch size allows for more context and image features within the patch to contribute towards QP estimation. Spatial strides were selected so that there was no patch overlap in the training set, although patches taken from the same video sequence would exhibit some correlation.

5.4.2 Network Architectures

For the purposes of this experiment, three simple network architectures were examined, summarised in Table 5.3. Image patches were format YUV 4:4:4, rescaled to values between 0 and 1 and whitened. In order to preserve compression artifacts in situ, no further data augmentation was used. Batch size was 128 patches. Unless otherwise noted in the results, network architectures 1 and 2 were implemented with stride = 2 for all convolutional and pooling layers.

Network architectures were designed with compression artifacts in mind. H.264/AVC uses a minimum DCT block size of 4x4 pixels so a 4x4 kernel aligns to this. Using an even-sized kernel is unusual but not without precedent [55]. A stride of 2 allows sufficient overlap to encounter artifacts while reducing the number of network parameters. Networks were trained and tested multiple times and average accuracy and confusion

Table 5.3: A summary of network architectures

Name	Layers
NA 1	conv4x4-64, pool3x3, norm, conv4x4-64, norm, pool3x3, fc-384, fc-192, softmax
NA 2	conv5x5-64, pool3x3, norm, conv5x5-64, norm, pool3x3, fc-384, fc-192, softmax
NA 3 [194]	conv3x3-32, conv3x3-32, pool2x2, conv3x3-64, conv3x3-64, pool2x2, conv3x3-128, conv3x3-128, pool2x2, conv3x3-256, conv3x3-256, pool2x2, conv3x3-512, conv3x3-512, pool2x2, fc-512, softmax

matrix values taken for the results.

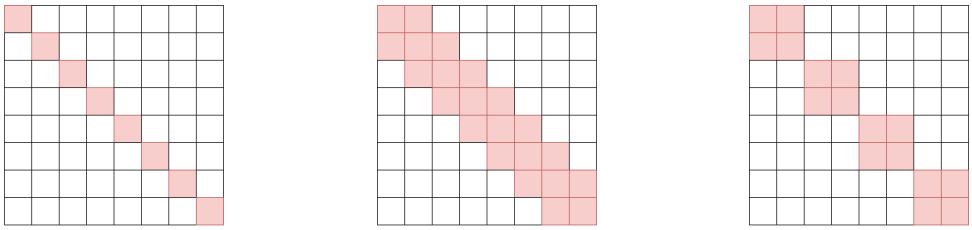
5.4.3 Estimating Quantisation Accuracy

The quantisation parameter (QP) in H.264/AVC can be expressed as:

$$0 \leq QP \leq 52,\ QP \in \mathbb{R} \tag{5.1}$$

QP relates directly to Q_s in Equation 2.1. Patches with similar QP labels exhibit similar compression features, and confusion matrices produced by the network reflected this. Two different QPs might have very similar effects on a given patch, depending on the patch content. An example of this is a whole patch of solid colour, which transforms to a single high amplitude, low frequency coefficient which is non-zero on quantisation. Such an extreme example is unlikely in natural scenes but it demonstrates how applying close QPs might result in identical patches with different labels. Therefore, QP has been sampled at [0, 7, 14, 21, 28, 35, 42, 49] in the synthesised datasets. Using all possible QP would also generate an extremely large dataset and increase model training times. Using a range of sampled QP, the confusion matrices produced by the model can be examined and super-classes composed to estimate accuracy. The sampled QP is sufficient to test the learned features to find out if they are useful. We leave more accurate QP estimation for future work.

Unlike the work presented in [199], where data was decomposed based on discrete classes, in this experiment, we combine each two adjacent classes into one, assuming that the change in pixels is not significant within adjacent QP. Figure 5.1 gives a visual demonstration of how this can be applied to a confusion matrix. In a confusion matrix, predicted labels from the network are tabulated against ground truth class labels. The overall accuracy of the network is given as the sum of the diagonal elements of the confusion matrix divided by the sum of all the elements (Fig. 5.1a). That is, the *correctly predicted* elements divided by all the elements. If some degree of error is



(a) Standard accuracy

(b) Combine adjacent classes

(c) Removing label ambiguity

Figure 5.1: Different class compositions in a confusion matrix

permitted in the confusion matrix, adjacent classes can be accepted as "correct" and a new accuracy $\hat{p}$ (Fig. 5.1b) can be calculated by combining adjacent classes thus:

$$\hat{p} = \frac{1}{M}\{\sum_{i=0}^{m} a_{i,i} + \sum_{i=1}^{m} a_{i-1,i} + \sum_{i=1}^{m} a_{i+1,i} + \sum_{i=1}^{m} a_{i,i-1} + \sum_{i=1}^{m} a_{i,i+1}\} \tag{5.2}$$

Although Equation 5.2 combines adjacent classes well in theory, testing how well it represents a model where classes are combined *before* training would involve assigning different labels to identical data. Instead, we compose super classes according to Equation 5.3 and Figure 5.1c.

$$\hat{p} = \frac{1}{M}\{\sum_{i=0}^{m/2} a_{2i,2i} + \sum_{i=0}^{m/2} a_{2i+1,2i} + \sum_{i=0}^{m/2} a_{2i,2i+1} + \sum_{i=0}^{m/2} a_{2i+1,2i+1}\} \tag{5.3}$$

Using Equation 5.3, labels can be unambiguously combined and a network trained on these labels. Equation 5.3 can also be extended to create even larger super-classes allowing for ever greater error.

5.5 Evaluating and Analysing Quantisation Estimation

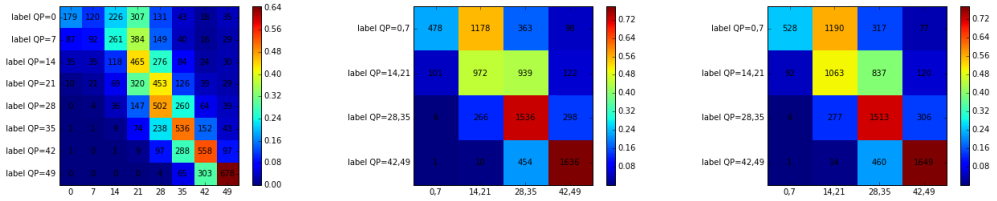
The initial experiment using CIFVid_80 achieved only 36.25% accuracy. Following [194], patch size was reduced and UCID was introduced, creating two new datasets: CIFVid_32 and UCID_32_large. The results in Table 5.4 show that smaller patch size did not improve accuracy, but training on intra frames only did. Halving the network stride parameter helped, but was still worse than using a larger patch size. Networks trained on UCID_32_large achieved accuracy of over 58% when tested with UCID_32_large test data, but approximately half that when tested with CIFVid_32.

CIFintra_32, comprising all key frames (Table 5.2) answered the question of why learning from UCID_32_large did not translate well to CIFVid_32. Patches in CIFVid_32 and CIFintra_32 come from exactly the same points in the video sequences, so their content is strongly visually correlated. Only the underlying compression modes differ. The best performing network architecture was re-tested with CIFintra_32. The accuracy on CIFintra_32 using the network trained on UCID_32 showed good improvement over testing on CIFVid_32 (54.14% vs 30.35%).

CIFVid_32, CIFintra_32 and UCID_32_large were mismatched in terms of patch quantity within the training sets. To investigate whether this accounted for some of the differences in accuracy, AllVid_32, AllIntra_32 and UCID_32 were created. Table 5.5 shows the accuracy achieved on each combination. The larger training set UCID_32_large improved accuracy by an average 7.9% on the UCID_32 test set but only average 2.34% on AllVid_32. The addition of extra training video patches when increasing CIFVid_32 to AllVid_32 did not increase accuracy. From these differences in performance, it can be concluded that the UCID-based datasets were less correlated than those derived from video sequences leading to more feature coverage and more generalisable networks. In Table 5.5 intra-only trained networks still out-performed those trained on AllVid_32, except in the case of AllIntra_32 versus AllVid_32. Given that AllVid_32 and AllIntra_32 contain visually similar pixel patches, this implies that the network trained on AllVid_32 has learned some features distinct to predicted frames that do not translate to key frames. This pattern was repeated with patch size 80x80.

Larger patch size datasets AllVid_80, AllIntra_80 and UCID_80 were generated and used to train and test different network architectures. Table 5.5 shows the results. Comparing results for AllVid_80 with CIFVid_80 and AllVid_32 (35.27%, 36.25%, 26.78%, respectively) show that increasing the number of patches did not improve accuracy but increasing the patch size did. Although networks trained and tested on UCID_80 achieved good accuracy (72.75%, Table 5.5), the learning did not transfer to AllVid_80 (36.72%). It did, however, translate to AllIntra_80 (63.40%). From this, it can be deduced that QP can be successfully estimated directly from the pixels of *key* frames but does not translate well to *predicted frames*. Networks trained on predicted frame patches achieve lower accuracy than that those trained on key frame patches. Moreover, the accuracy of networks trained on UCID_80 is higher than those trained on AllIntra_80. This can be partly attributed to weaker correlation in UCID_80 image patches.

Overall, NA 3, the deepest network, had the lowest accuracy. The limited size of the datasets may have contributed to this, but it is more likely that the depth of the network did not help with compression features. Compression features in H.264/AVC



(a) Full (35.27%) (b) Reduced (54.65%) (c) Composed (56.25%)

Figure 5.2: Confusion matrices for NA 2 trained/tested on AllVid_80 (overall accuracy for a single network)

are related to the size of the transforms used in the codec and these vary from 4x4 to 16x16 pixels. It can be deduced that though deeper networks go some way to accounting for differences in scale in traditional object classification, this is largely unnecessary when examining compression.

Table 5.4: Initial results: Accuracy for patch size 32 (network 3 failed to train)

Network	Tested on	CIFVid_32 trained	UCID_32_large trained
1	CIFVid_32	27.30	33.14
1	UCID_32_large	31.00	58.55
2	CIFVid_32	26.69	30.35
2	UCID_32_large	32.23	59.28
2	CIFintra_32	27.30	54.14
2 (stride=1)	CIFVid_32	25.34	33.67
2 (stride=1)	UCID_32_large	28.46	66.88

5.5.1 Relaxing the Problem

Although the overall accuracy achieved on a network trained on predicted frame patches from AllVid_80 was low, the confusion matrix implied a reasonable error rate. Figure 5.2a shows the confusion matrix for NA 2 trained/tested on AllVid_80. Average accuracy was 35.27%. With class labels combined as in equation 5.3, the accuracy estimated from the confusion matrix is 54.65%. A network trained on the reduced label dataset yields a comparable accuracy of 56.25%. Therefore, the results obtained from calculations on the confusion matrix after training are comparable to networks trained specifically on these super classes. Table 5.6 shows that this is true across all patch size 80 datasets. Composing super classes from adjacent classes prior to training reduces the number of labels and slightly enhances generalisation in the network, yielding slightly higher results from video-based datasets with correlation between video patches. The same pattern was repeated with other architectures. QP

Table 5.5: Cross evaluation on similar sized datasets. accuracy for patch size 32 (network 3 failed to train) and for patch size 80

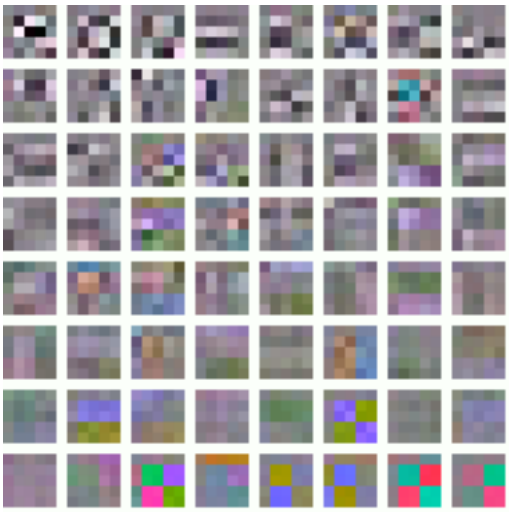
	Patch size 32				Patch size 80			
Network	Tested on	AllVid_32 trained	AllIntra_32 trained	UCID_32 trained	Tested on	AllVid_80 trained	AllIntra_80 trained	UCID_80 trained
1	AllVid_32	26.72	24.68	31.34	AllVid_80	34.93	26.02	36.72
1	AllIntra_32	27.75	33.75	44.14	AllIntra_80	34.11	37.94	63.40
1	UCID_32	33.74	41.56	50.96	UCID_80	41.28	47.46	72.75
2	AllVid_32	26.78	25.38	31.79	AllVid_80	35.27	29.39	37.28
2	AllIntra_32	27.07	33.16	45.05	AllIntra_80	34.93	46.54	62.50
2	UCID_32	33.83	41.25	51.07	UCID_80	42.04	56.66	71.65
3	-	-	-	-	AllVid_80	29.97	24.91	29.35
3	-	-	-	-	AllIntra_80	29.94	42.67	55.95
3	-	-	-	-	UCID_80	38.99	53.81	61.17

in predicted frames can be estimated to within ± 7 (one class) with more than 54% accuracy. Higher quality frames are more challenging and this may be attributed to the larger range of frequencies available in uncompressed data. CNN models cannot distinguish between frequencies removed by compression and those simply absent in the source data.

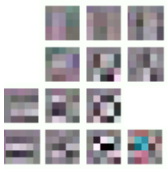
Table 5.6: Accuracy for patch size 80 (NA 2): composition after/before training

Tested on	AllVid_80 trained	AllIntra_80 trained	UCID_80 trained
AllVid_80	54.65 / 56.25	52.99 / 54.08	60.70 / 59.55
AllIntra_80	55.24 / 56.51	66.62 / 67.55	78.81 / 78.47
UCID_80	66.81 / 67.94	78.58 / 79.57	88.43 / 88.41

The shape of the confusion matrices (Fig. 5.2) gives insight into the model's learning. Confusion matrices across all architectures and datasets demonstrated similar shapes where the bottom left corner approached zero. Patches of high QP were seldom misclassified as low QP. In contrast, the top right corner of the confusion matrix, although it displays lower numbers than the diagonal portion, does not always approach zero. This pattern suggests that the model has learned something about the frequency domain. Natural images contain a large variety of frequencies, however quantisation in the frequency domain selectively reduces these. Weaker (low amplitude) frequency components are quantised to zero and thus filtered out of an image, leaving behind only dominant frequencies. For natural scenes, where lower frequencies tend to dominate, quantisation applied in video compression is effectively a low pass filter. This explains the blocky appearance of compressed video. Low amplitude, high frequency components aid smooth colour or intensity transition and removing these components leads to sharper colour transitions at macroblock edges. The detection of high frequency components within macroblocks therefore indicates lower QP. Unfortunately, the converse is not necessarily true. An absence of high frequencies does not indicate high QP, since some images naturally lack high frequency components.



(a) First Layer Filters (Network Architecture 2, AllIntra_32



(b) First layer filters

(c) 4x4 DCT

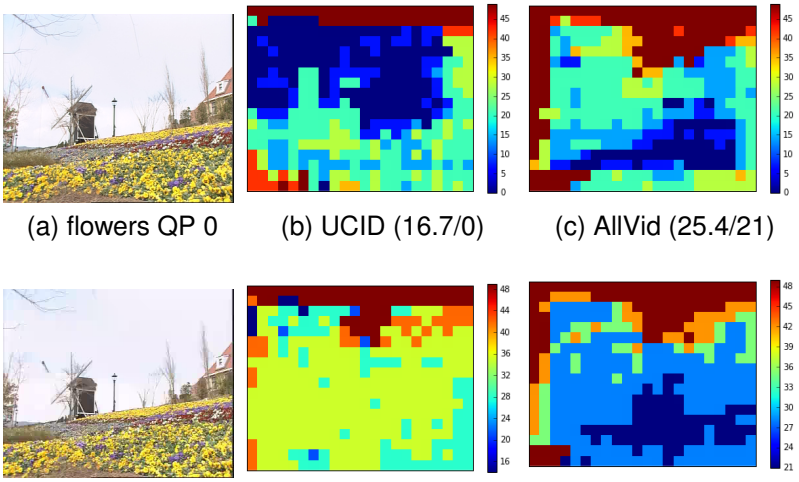
Figure 5.3: Some of the first layer filters display visual similarity to a spatial representation of the 4x4 DCT transform used in H.264/AVC. The first layer filters in this image are 5x5 pixels, enlarged to enhance visibility.

5.5.2 First Layer Filters

A visual examination of the first layer filters confirms the presence of frequency features. Figure 5.3a shows that these are not typical edge features and colour blobs like those associated with image classification. Figure 5.3c shows a spatial representation of the 4x4 Discrete Cosine Transform used in H.264/AVC. These are pixels that result from an inverse DCT on a 4x4 coefficient matrix where only one coefficient is non-zero. Figure 5.3b shows selected first layer filters from NA 2 trained on AllIntra_32. Visual similarities are obvious. The CNN uses some first layer filters to infer frequency information directly from the pixels. Statistical analysis of frequency was also used in [197] to detect multiple compression.

5.5.3 Whole Image Heat Map

Figure 5.4 shows classification results for 80x80 patches of a key frame from the sequence “flowers” at QP 0 and 35. A plain black border was added added after compression to allow classification of 80x80 patches centred on every 16x16 macroblock. The heatmaps all show misclassification along the top row due to the black border. Comparing Figures 5.4a and 5.4d, it is difficult for human eyes to differentiate between QP values, despite the large difference. The fine colour transition in the sky section is correctly classified by a network trained on UCID_80 as low QP. The network trained on AllVid_80 tends to overestimate sky QP and perform better on the colourful flower section. At moderate QP, the model trained on UCID_80 performs well. The AllVid_80 trained network achieves a reasonable mean over the whole image but underestimates QP in busy areas and overestimates in areas with fewer sharp edges. Mode 28 in Figure 5.4f shows how the model confuses adjacent labels and validates the use of superclasses.



(a) flowers QP 0 (b) UCID (16.7/0) (c) AllVid (25.4/21)

(d) flowers QP 35 (e) UCID (35.6/35) (f) AllVid (33.0/28)

*Figure 5.4: The heat maps showing the predicted QP for a frame from the sequence “flowers”, NA 2 used **(mean/mode)***

5.5.4 Limitations

It should be noted that although it was possible to train many standard layer-style CNNs to estimate the quantisation parameter, successful training was highly susceptible to initial conditions. Some networks with the given architecture failed to train or trained only after a large number of epochs. This may support the authors of [18], who argued that convolutional neural networks in their current form are better suited

to learning features relating to image *content* rather than those associated with manipulation detection. However, it can be seen in this chapter that some networks can be trained on this data and they learn frequency-style features consistent with the compression algorithm used. Moreover, some applications of deep neural networks, such as those mentioned in Section 3.3 show that many neural networks have an almost unreasonable affinity for detecting patterns in images that are invisible to human eyes. These patterns may be related to compression, and in order to investigate this, it is reasonable to use a deep architecture with no novel layers. Although novel layers have been proposed to detect low level features associated with video manipulation [18], this work demonstrates that they are not necessary for *all* low level video features.

Additional experiments performed with closer together QP, such as QP = [0,1,2,3,4,5,6,7]; QP = [0,2,4,6,8,10,12,14] and QP = [0,3,6,9,12,15,18,21] demonstrated that accuracy was much lower when QP was closer together. In each case, a perfectly balanced dataset was prepared using only intra frames and 80x80 patch size. In the case of QP = [0,1,2,3,4,5,6,7], the network trained but achieved accuracy little better than random chance. For QP = [0,2,4,6,8,10,12,14], the best accuracy achieved was 16.37%. For QP = [0,3,6,9,12,15,18,21], the best accuracy was 21.64%. It can be the case in compression that applying different QP to the same pixels can result in identically transformed pixels: the different frequencies removed by the quantisation process simply do not exist in the data, so there is no change. The simplest example of this is an area of flat colour which will remain unchanged by compression, regardless of the QP value. In the case of our datasets, this could potentially result in pixel patches which consisted of identical pixels but differing labels. To analyse how frequent this was in natural content, the datasets were checked to see how many duplicates they contained. It was found that less than 1% of the patches in the dataset were exactly duplicated in pixels but not in label, and that duplication was limited to the test set or to very low QP values. This is acceptable and expected in a problem which is intrinsically ambiguous. However, this test did not cover near-duplication and it can be hypothesised that the image patches were simply too similar to allow the network to differentiate between them. The strength of CNNs is in their ability to generalise. Data augmentation aids learning in CNNs specifically because it allows more generalised features to be learned. It can be hypothesised that if a process creates effects that are more subtle than those produced by common data augmentation, then it may not be possible for a standard CNN to adequately learn these effects. In any case, the dataset with QP=[0,7,14,21,28,35,42,49] was used moving forward because, although not perfect, it may be *good enough* for the application.

5.6 Conclusions

We have shown that the level of compression of small image patches can be estimated objectively by a CNN with a relatively straightforward architecture. The accuracy of CNNs trained on intra frames is much higher than those trained on predicted frames. The experimental results also strongly suggest that compression features learned from still images (intra frames) alone do not transfer to predicted frames. Although a neural network can be trained to estimate the QP in key frame patches, the results for predicted frames were weaker. In predicted frames, quantisation is applied to the residual difference between predicted and actual pixels. CNN compression estimation may be improved by using residuals from the compressed bitstream rather than reconstructed pixels. It may also be necessary to first identify intra macroblocks within an image or intra frames within a sequence in order to gain an estimate of accuracy in QP estimation and use this feature to its maximum advantage.

Larger patch sizes yield higher precision but further investigation is needed to clarify whether an optimum patch size exists. Smaller patch sizes are desirable if the model is to serve as accurate tampering detection. The features learned in the first layer of CNNs strongly resemble patterns of DCT coefficients, so QP estimation or the reliability of network training may be improved by initialising some of the first layer weights with DCT patterns or moving the problem in its entirety to the frequency domain.

Compression is no longer an irreversible “black box”. Although recompression destroys information about original compression mechanisms in the compressed bitstream, tell-tale signs in the pixels can be used to estimate original levels and this information could be used to detect video tampering, including splicing and multiple compression.

Chapter 6

A Compression Fingerprint

Chapter 5 examined learning quantisation parameters from pixels in detail, but found a number of factors that influence the accuracy of this. In this chapter, we look at how these factors can be detected and thus offset any disadvantages. In particular, it was found in Chapter 5 that QP estimation from pixels by CNN was more accurate for intra frames than predicted frames. Therefore, this chapter focusses on how to determine from pixels which frames are intra frames and which are predicted. To this end, we examine learning the inter/intra mode directly from pixel patches using CNN.

We also look at learning the H264/AVC deblocking filter [200, 27] setting directly from pixels. The deblocking filter in H.264/AVC is a psychovisual feature which filters pixels at the edges of macroblocks and sub-blocks according to the size of the associated motion vector. It counteracts block artifacts in a sequence. It is turned on or off at a sequence level and, because of this, could prove a useful feature for detection of spliced video.

The main findings of this chapter and Chapter 7 are accepted to appear in the journal "Neural Computing and Applications" as the paper "Video Tampering Localisation Using Features Learned from Authentic Content" [26].

6.1 Pixel Forensics

Any processing, particularly compression, can leave a forensic fingerprint on the pixels of a video sequence. Analysing this fingerprint can provide a means to detect evidence of video tampering, such as splicing, inpainting or inter-frame tampering. Here we look

specifically at aspects of video compression that can usefully contribute towards this fingerprint.

Fingerprints in video have been examined before, primarily in the area of video forensics. Sensor pattern noise for camera model identification was examined in [1]. Shullani et al [20] found that sensor pattern noise was affected by the compression applied by two different social media platforms. However they also showed that video processed by YouTube could be differentiated from video processed by WhatsApp. This implies that individual social media platforms use different encoder settings, or even encoders, which leave evidence of themselves in the pixels. In [167], an ensemble of CNNs was used to extract the processing history of a given image, including evidence of resizing, high pass filtering, recompression and Gaussian blurring. Unlike [18], the authors of [167] did not use any novel convolutional layers. This shows that evidence of processing can be learned by ordinary CNN layers.

As noted in Section 2.1, in H.264/AVC and MPEG-2, frames divide into macroblocks and each macroblock can be intra or inter coded. Intra frames can only contain intra macroblocks, but inter frames can contain both intra and inter macroblocks. For non-predicted data, the pixel data itself is transformed into the frequency domain using Discrete Cosine Transforms (DCTs), quantised and variable length encoded for transmission. For predicted data, a suitable patch of reference pixels is located, then the *difference* between current and reference data is transformed, quantised and encoded. Knowledge of the inner workings of a compression encoder shows that the inter/intra decision is made at the block level.

The detection of inter/intra mode is important in video tampering detection. There are already a number of techniques for detecting video tampering which utilise intra-frame detection. The authors of [201, 129] noted that recompressing a key frame as a predicted frame resulted in statistical features in the compression choices made by the encoder; specifically, former intra frames had fewer skipped macroblocks when re-encoded as predicted frames. These patterns were used to detect recompression and could also detect deleted frames, provided a fixed GOP was used. The method proposed in [202] also used intra-frame position analysis to detect digital video forgeries. Using machine learning techniques, deleted frames in an MPEG-2 encoded video sequence were detected with 95% accuracy using elements of compression. However, given that compression features were extracted directly from the bitstream itself, the methods of [202] could be simply defeated via recompression. In this chapter, we aim to overcome that challenge by estimating compression parameters directly from the pixels. In Chapter 7, these patterns are then used to identify areas of inconsistency

related to video tampering. In their survey, the authors of [21] noted that many inter-frame tampering detectors failed when dynamic GOP was used or when an integer number of GOPs were removed, implying that their functionality was based on identifying "misplaced" intra-frames. This research shows that identification of intra-frames in itself is a useful feature in video manipulation detection.

While intra-*frame* detection is useful in inter-frame tampering and recompression detection, detecting a patch of intra-*macroblocks* could be useful for detecting splicing or spatio-temporal copy-moves. There is no guarantee that intra-frames will align when compositing two video sequences together. Intra-frames occur relatively infrequently, so it is actually statistically unlikely for the intra-frames of composited content to co-occur. Therefore, detection of intra-regions is a valid feature in itself to use to detect tampering. There is also the possibility of some encoders using periodic intra refresh, whereby intra-frames are only sent at the very beginning of a sequence. Subsequent access points into the sequence are provided by intra-encoding successive regions of macroblocks. Motion estimation is then restricted to use only refreshed regions, and the decoder instructed to postpone frame display until a full frame has been refreshed. This mode of operation has the potential to nullify any method that relies on intra-frame detection, however intra-region detection may provide new valid methods.

The H.264/AVC deblocking filter is a filter designed to remove blocking artifacts as part of the encoding process. Blocking artifacts (see Section 2.1.6) arise due to the use of macroblocks. The deblocking filter acts at the horizontal and vertical edges of blocks. Macroblock edges are filtered according to different parameters defined in the standard [27], such as the size of the motion vectors and the deblocking filter settings. It is one of the more visible compression settings. It is set on a sequence level and, as such, provides a theoretical means to identify regions which have come from different sequences and thus detect splicing.

6.2 Learning Compression Features

As noted in previous sections, there are a number of video features related to compression that are likely to be useful in video analysis. We will examine:

- Quantisation Parameter (QP), which will give a measure of compression
- Inter/intra (I/P) frame mode, which will help to ascertain the expected accuracy of the QP estimate

- Deblocking filter settings, which may help to differentiate between the pixels of sequences with different psycho-visual settings
- Frame deltas, or the differences between frames
- Key frames, inferred from the other features

The methods used to identify these features are detailed in the following subsections. QP, I/P and deblocking features are learned from authentic sequences by CNN. The frame delta value was set to 1 whenever the mean absolute difference of a given 16x16 pixel patch and the co-located patch in the previous frame was non-zero, and set to zero otherwise.

6.2.1 Authentic Datasets for CNN training

When examining the effects of compression, is vital to start with unprocessed data. For this we used standard YUV 4:2:0 sequences from xiph.org, which are commonly used for video compression quality analysis [37]. Strictly speaking, YUV 4:2:0 is a compressed format due to reduced resolution of the colour channels however it is widely used as a starting format in video compression. The sequences from xiph.org come in various dimensions and cover a wide variety of subjects from studio-shot sequences to outdoor scenes. All sequences are single camera, continuous scenes with varying degrees of camera motion.

A large amount of data is required to train a neural network and uncorrelated data will produce a more generalised network. It is possible to use still image data as single frame sequences when focussing on spatial compression artifacts and excluding temporal compression. For this purpose, the images of UCID [166] were used. UCID consists of uncompressed images which are either 512x384 pixels or 384x512 pixels and cover a wide variety of subject matter. All are natural scenes and taken with the same camera. Of the original reported 1338 images in the dataset, only 882 were available for download[1]. Using a dataset of single images is not ideal since predicted frames cannot be examined. However it allows for a greater variety of pixel combinations in a smaller dataset because individual images are uncorrelated. Each image from UCID was regarded as a single frame video sequence.

Following [25], we processed the video using various compression parameters to synthesise a number of original datasets summarised in Table 6.1. Each video sequence was compressed using the open source H.264/AVC encoder x264 and one of a range

[1] UCID images from http://jasoncantarella.com/downloads/ucid.v2.tar.gz

Table 6.1: A summary of original datasets from publicly available sources

Name	Source	Length	Dimensions	Key	Deblock
AllVid	xiph.org	45 videos	176x144 to 1920x1080	1/250	off
AllIntra	xiph.org	45 videos	176x144 to 1920x1080	all	off
AllDeblock	xiph.org	45 videos	176x144 to 1920x1080	1/250	on
UCID	UCID [166]	882 images	512x384 or 384x512	all	off

of constant QP levels in variable bitrate mode. Constant quantisation parameters were selected with an even distribution: QP=[0, 7, 14, 21, 28, 35, 42, 49]. Constant bitrate rate control and psychovisual options were turned off. Deblocking filter was set as specified in Table 6.1. For datasets containing predicted frames, the key frame interval was 250. Patches were then extracted from the decoded YUV 4:2:0 sequences. Patches were upsampled from YUV 4:2:0 to YUV 4:4:4, where the Y-channel represents intensity and U and V channels are colour.

Table 6.2 summarises synthesised datasets. A large temporal stride was used to limit correlation between patches from the same video sequence. Consecutive frames are similar to each other and a neural network trained with correlated dataset will be subject to overfitting. All datasets were prepared in advance of network training and original video sequences were split into disjoint train and test sets prior to compression and patch sampling to prevent data leakage[2]. The images of UCID were encoded as all intra frames, and used as supplemental data to AllIntra.

A patch size of 80x80 pixels was used. Block edge artifacts in intra frames will present themselves at macroblock and sub-block boundaries. Therefore, any patch size larger than 16x16 will capture block edge artifacts. In [194], a small patch size of 32x32 was selected, but results in [25] and Chapter 5 showed that larger patch sizes yielded more accurate local results. When aligned with the macroblock grid, 80x80 pixels covers 5x5 complete macroblocks. A larger patch size allows for more context and image features within the patch to contribute towards QP estimation. Spatial strides were selected so that there was no patch overlap in the training set, although patches taken from the same video sequence would exhibit some correlation.

Each dataset in Table 6.2 consists of a number of YUV 4:4:4 patches, each labelled appropriately. QP was labelled according to the quantisation parameter. The inter or intra labels depended only on the frame type, and not on individual macroblock types. The deblocking filter label was selected based on sequence level parameters. From each synthesised dataset, a neural network was trained to perform classification

[2]Training sequences: akiyo, bridge-close, bridge-far, carphone, claire, coastguard, foreman, hall, highway, mobile, mother-daughter, paris, silent, stefan, tennis, waterfall, old_town_cross, crowd_run, ducks_take_off, in_to_tree, mobcal, old_town_cross, parkrun, shields. Test sequences: bus, flower, news, tempete

Table 6.2: A summary of patch datasets

Name	Source	Label	# Train	# Test
IntraForQP	AllIntra	QP	764640	56392
Intra0vsInter1	AllIntra, AllVid	I=0, P=1	836512	12992
Deblock1	AllDeblock, AllVid	Deblock=0,1	836512	12992

according to the label.

6.2.2 Network Architecture

In Chapter 5, three different network architectures (NAs) were examined for one compression parameter (the quantisation parameter or QP). Here, one fully convolutional architecture was used, similar to the architecture used in [159] which obtained particularly good results on CASIA2 [44]. CASIA2 is known to suffer from asymmetric image processing between tampered and non-tampered image classes [160], therefore this network architecture is already known to perform well in detecting image processing, and specifically recompression. An independent network with this architecture was trained for each separate compression parameter, yielding three sets of network weights, one each for QP, deblock and inter/intra.

Using the notation defined in Table 2.3, the architecture used was: conv5x5-30, norm, pool2x2, conv3x3-16, conv3x3-16, conv3x3-16, norm, pool2x2, conv3x3-16, conv3x3-16, softmax. A convolutional stride of 2 allowed sufficient overlap to encounter compression artifacts while reducing the number of network parameters. Input was image patches of format YUV 4:4:4 which were rescaled to values between 0 and 1 and whitened. In order to preserve compression artifacts in situ, no further data augmentation was used. Batch size was 128 patches. Adam [203] was used for gradient descent in the quantisation parameter network and stochastic gradient descent for the intra/inter and deblock features. The networks were implemented using TensorFlow.

6.2.3 CNN Compression Parameter Estimation Accuracy

The mechanism of quantisation with H.264/AVC is discussed in Section 2.1.1 The quantisation parameter (QP) in H.264/AVC can be expressed as:

$$0 \leq \text{QP} \leq 52, \ \text{QP} \in \mathbb{R} \tag{6.1}$$

Patches with similar QP labels exhibit similar compression features, and confusion

matrices and analysis in Section 5.5 reflect this. Two different QPs might have very similar effects on a given patch, depending on the patch content. As noted in Section 5.5.4, pixel-wise duplicates are uncommon, but using close QP results in low network accuracy. Therefore, QP was sampled at [0, 7, 14, 21, 28, 35, 42, 49]. Another source of ambiguity is the presences of skipped macroblocks. In simple terms, a skipped macroblock in a predicted frame can be almost identical to the reference macroblock in the reference frame. This means that there may exist some predicted regions whose pixel content is identical to regions in a key frame. Using larger patch sizes decreases this risk, and, as noted in Section 5.5.4, the datasets contained very few samples where pixel content was identical but labels differed.

6.2.4 Key Frame Detection

One thing that was evident from Section 5.5 was that accurate estimation of quantisation parameters in predicted frames is challenging. This is because the quantisation parameter is applied to the *difference* between the motion compensated macroblock from previous encoded frames and the current macroblock being encoded. If this delta is unknown, as in the case where only the pixels can be relied on, then it is difficult to estimate the quantisation parameter. For this reason, QP estimation on predicted frames relies on the presence of skipped blocks which remain unchanged since the last key frame and intra blocks. In order to avoid such challenges, it was decided to identify and process only key frames.

A large percentage of compression in video comes from predicted frames. It is much more efficient to compress the differences between frames than it is to compress every single frame independently. With the advent of integer transforms in standard compression codecs, periodic key frames are no longer required to correct transform rounding error accumulation. Therefore, it is reasonable to assume that key frames are in the minority in a video sequence. Moreover, because non-predicted frames are inherently larger than predicted frames, and rate control mechanisms attempt to avoid peaks in bitrates, key frames are often compressed using higher QP than predicted frames. Key frames can also exhibit more block artifacts than predicted frames since macroblock edges are all aligned. These features can be used to identify key frames.

To identify key frames we first estimated the quantisation parameter qp, the inter/intra parameter ip and the deblocking parameter db for patches in every frame of a sequence. Note that qp is distinct from QP in that the variable qp has been estimated directly from the pixels themselves as, as such, may be subject to inaccuracies. The patches were 80x80 pixels and separated by a stride of 16 pixels (overlapping). Patch

values for qp, ip and deblock were then averaged over each frame in a sequence and the differences between the averages taken. The three different predictions were then combined as in Equation 6.2

$$a_f = (\overline{qp}_f - \overline{qp}_{f-1}) \times (\overline{ip}_f - \overline{ip}_{f-1}) \times (\overline{db}_f - \overline{db}_{f-1}) \tag{6.2}$$

Where $\overline{qp}_f$ represents the average CNN predicted quantisation parameter for frame f and $\overline{qp}_{f-1}$ is the same parameter for the previous frame. Key frames were then defined as any frame where the value of a_f was more than two standard deviations from the mean of a_f.

6.2.5 Tampered Video Datasets

Two publicly available datasets were used for evaluation: D'Avino et al [16] and Video Tampering Dataset (VTD) [15]. Both of these datasets are intended for tampering detection, but contain evidence of real world compression. VTD is distributed via YouTube and, as such, has also been subject to recompression. The details of the latest YouTube compression are available direct from the downloaded bitstream. D'Avino's dataset is supplied as uncompressed .avi files, therefore there is no bitstream evidence of compression, however the video sources are well described in the associated paper [15] and some compression features can be inferred. Further details of these datasets are available in Section 7.3.

Raw YUV 4:2:0 files from Derf's media collection [37] were also used to test the efficacy of the intra-frame detection (see Section 6.3.2 for details on processing and results).

6.3 Evaluation of Compression Feature Detectors

Results indicate that standard CNNs can be trained to achieve a reasonable level of accuracy in determining three compression parameters directly from pixels, and that this accuracy is sufficient to identify key frames.

Table 6.3: Accuracy of CNNs trained for compression parameter classification

Trained to classify	Number of classes	Accuracy
QP	8	71.18 %
Inter/Intra	2	69.23 %
Deblock	2	66.53 %

6.3.1 CNN Compression Parameter Estimation

The three trained CNNs achieved the accuracies listed in Table 6.3. Training a network to detect compression parameters directly from pixels will always be subject to some degree of error. For example, areas of flat colour, although relatively uncommon in natural images, are not subject to quantisation of frequency components, since there are no high frequency components to quantise. This is an extreme example, but it can be generalised: not all pixel regions will naturally contain all relevant features necessary for classification. This effect can be seen in the confusion matrix (Figure 6.1). The network incorrectly labels some lightly compressed patches as heavily compressed, and this is likely due to a natural lack of high frequencies in those regions. Similarly for the inter/intra network, we label on the frame level, not on the macroblock level. It is possible for the pixels in a macroblock to remain unchanged between key and predicted frames if the video content is static. With regards to the deblock filter, although it is set on or off on a sequence level, the parameters which control the level of filtering on individual macroblocks are also partly controlled by motion vectors. All of these factors contribute to ambiguity in the labels for individual macroblocks, but are sufficiently evened out over a patch size of 80x80 pixels to achieve a workable level of accuracy. This can be seen in Figure 6.2 where the average estimated QP for the complete frame is accurate but individual regions display some variance. In particular, the white sky region is allocated a relatively high QP, demonstrating a lack of high frequency coefficients in this saturated area.

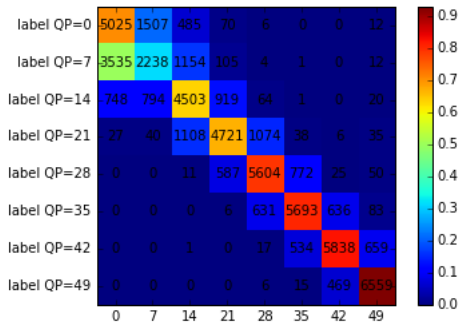


Figure 6.1: Confusion matrix for QP-trained network

Figure 6.2: QP heatmap for test sequence flowers, QP=35. The heatmap gives an average prediction of QP=35 but there is some variation between individual regions

6.3.2 Key Frame Identification

The key frame identification was performed as in Section 6.2.4. Because it is based on outliers, this method of key frame identification assumes at least one key frame in the sequence, other than the initial frame. Key frames occur when specified by the compression encoder. Cut scenes will sometimes trigger the encoding of a key frame, but not always, and not all key frames occur on cut scenes. Frame differences can sometimes indicate key frames, but they are not reliable as can be seen by comparing Figures 6.3 and 6.4.

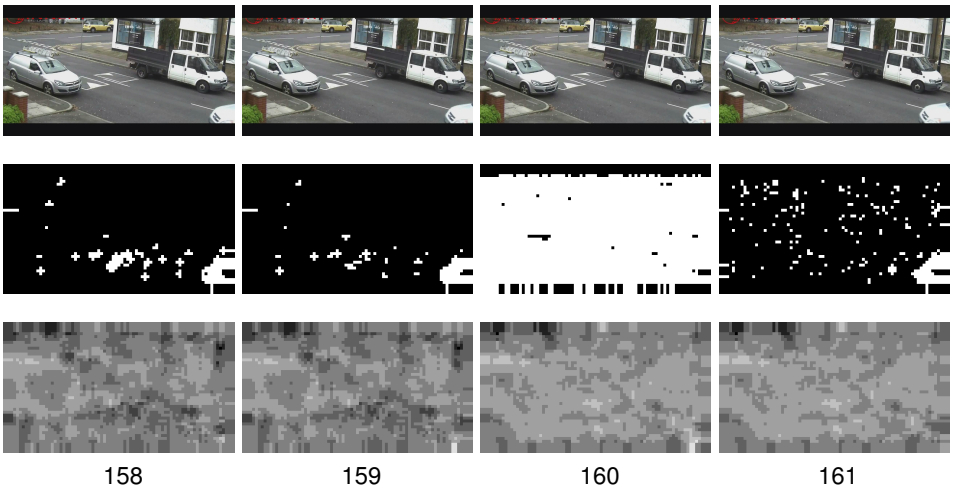


Figure 6.3: Frames 158-161 of sequence "forgery CCTV_London_Str" [15], showing (top to bottom) sequence, binary frame difference for 16x16 blocks (black = no difference, white = differences) and QP prediction using a trained neural network. Frame differences clearly indicate the key frame, even though it is not visible in the sequence. The key frame is frame 160

To gauge its efficacy, the method of key frame identification was tested on a number of sequences which comprised compressed and recompressed versions of YUV test sequences. These were first compressed with the open source encoder x264 using different bitrates (0.01, 0.02, 0.05, 0.1, 0.2 bits per pixel) and an intra-frame frequency

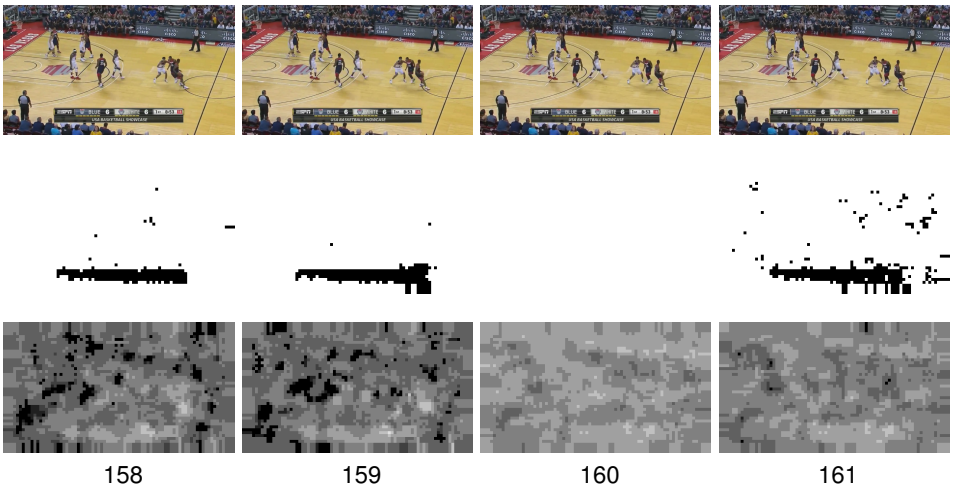


Figure 6.4: Frames 158-161 of sequence "forgery basketball skills" [15], showing (top to bottom) original sequence, binary frame difference for 16x16 blocks (black = no difference, white = differences) and QP prediction using a trained neural network. Frame differences do not always highlight key frames. The key frame is frame 160

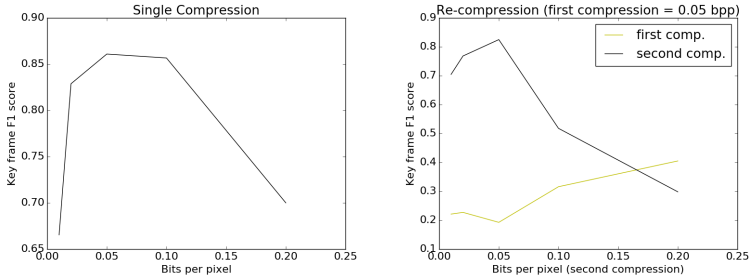


Figure 6.5: F1 scores for key frame identification in singly and doubly compressed sequences

of 1/30. The resulting compressed sequences were then re-compressed using the same bitrates but an intra-frame frequency of 1/25. It was found that the method performed very well between bitrates of 0.02 and 0.2 bits per pixel (bpp) in identifying the key frame from the latest compression. This is shown by the graph for single compression and the line for the second compression in the graph of double compression in Figure 6.5. Below bitrates of 0.02 bpp, the visual video quality was very poor and the predicted quantisation parameter started to saturate to its highest level, leading to inaccuracies in key frame identification. Above bitrates of 0.2 bpp, the predicted quantisation parameter did not saturate but accuracy still dropped. It is probable that the reduced accuracy was due to rate control choices made in the x264 encoder. With a higher bitrate available, peaks in bitrate due to key frames are comparatively reduced. Therefore, it becomes more efficient to encode key frames with higher quality, yielding

more accurate reference frames and consequently reducing the bits required for predicted frames. Re-compression at bitrates below 0.1 bpp effectively camouflaged key frames from the previous compression. As bitrates of the second compression process increased, evidence of key frames from the previous compression process emerged, as can be seen by the rise in the "first compression i-frames" F1 score graph.

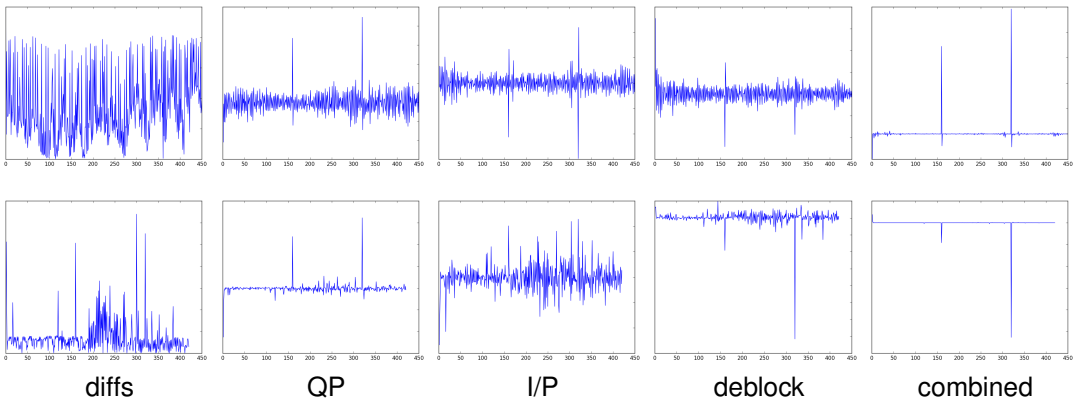


Figure 6.6: Graphs showing differences between the mean value per frame of each feature for the sequences "Forgery basketball skills" (top) and "Forgery CCTV London Str" (bottom) [15]. Frame averages vs frame number for (left to right): ***absolute frame differences; predicted QP; predicted inter/intra; predicted deblock; combination as in Equation 6.2****. The key frames can be clearly identified as outliers using a combination of the CNN predicted features and verified as correct by analysing the compressed bitstream.*

The method of identifying key frames was then applied to VTD. It should be noted that our method was effective at bitrates corresponding to those of VTD. As can be seen in the graphs in Figure 6.6, combining predicted QP, inter/intra and deblocking values as in Equation 6.2 provided a clear indication of key frames in the latest compression. Using the frame averaged mean absolute difference between frames yielded noisy results and did not accurately identify key frames. Comparing the key frames identified using this method with those extracted from the bitstreams of the forged sequences of VTD [15] achieved 87 true positives out of a total of 93 key frames. There were 27 false positives, giving an F1 score of 0.84. The majority of false positives (16 false positives) came from two sequences: "Forgery cake cooking" and "Forgery Awesome Cuponk" which both contain "fade" cut scenes. "Forgery cake cooking" also contains evidence of temporal upsampling from 25 fps to 30 fps. The robustness of this method of key frame detection against temporal upsampling has not been investigated, and this is left for future work. Since non-uniform temporal upsampling would be indicative of video splicing, a method to detect it would prove useful.

The key frame detection method was also applied to D'Avino et al's dataset [16]. Although the dataset is supplied as uncompressed .avi files, the key frame detector

provided evidence of previous compression by locating regular key frames at approximately 30 frame intervals.

6.4 Conclusions

We have shown that three features of H.264/AVC compression, namely quantisation parameter, intra/inter and deblock modes, can be estimated objectively by CNN using authentic, synthesised datasets. These features have been used to predict the location of key frames in video sequences, where they provide some advantage over the use of simple frame deltas. We have also investigated how well the intra-frame localisation works through recompression and shown that evidence of previous compressions remain provided subsequent compressions use a sufficient bitrate or quality level. Previous compression can be effectively laundered only when subsequent recompressions use lower quality encoder settings, such as lower bitrate. This is consistent with the functionality of H.264/AVC compression.

Chapter 7

Tampering Detection

Chapters 5 and 6 looked at how the compression features of a sequence can be estimated and analysed. Here, we examine how these features can be used in detection of video manipulation.

Video tampering detection remains an open problem in the field of digital media forensics. As video manipulation techniques advance, it becomes easier for tamperers to create convincing forgeries that can fool human eyes. Deep learning methods have already shown great promise in discovering effective features from data, particularly in the image domain, however they are exceptionally data hungry. Labelled datasets of varied, state-of-the-art, tampered video which are large enough to facilitate machine learning of state-of-the-art tampering techniques may never exist while the field of digital video manipulation is advancing at such an unprecedented pace. Therefore, it is vital to develop techniques which can be trained on authentic or synthesised video but used to localise the patterns of manipulation within tampered videos. In this chapter, we develop a framework for tampering detection which uses the features derived from authentic content in Chapters 5 and 6 to localise tampered regions in publicly available tampered video datasets. Extensive evaluation suggests these features can be used successfully to localise tampering, subject to some explainable limitations.

The main findings of this chapter and Chapter 6 are accepted to appear in the journal “Neural Computing and Applications” as the paper “Video Tampering Localisation Using Features Learned from Authentic Content” [26].

7.1 Challenges in Detecting Tampered Video

There are a number of challenges in the detection of tampered video. As shown in Chapter 3 there are many, many different ways to alter video content, but comparitively few publicly available datasets. In their 2017 paper reviewing video content authentication techniques, Singh and Aggarwal [70] noted that there was no consistent database of realistically doctored videos. More recently, a number of datasets have become available, but their composition varies. VTD [15] supplies authentic and tampered versions of the same sequences but does not explicitly supply masks. D'Avino et al [16] supplies tampered and authentic counterparts and includes masks. FaceForensics [17] focused exclusively on the tampering method of Face2Face [6], but its recent successor FaceForensics++ [22], also provides video that uses FaceSwap and DeepFakes manipulation.

Some video manipulation techniques are completely invisible to human viewers, so a fully labelled dataset is required. Manipulation techniques are currently more powerful than detection techniques [168], with many ways to digitally alter an image or video but relatively few methods to detect such manipulations, and fewer still that are agnostic to the type of tampering applied. There is therefore a need to develop detection techniques that are independent of the type of video manipulation. Moreover, some video manipulation methods ultimately create a video sequence where the majority of pixels are authentic. This leads to a class imbalance problem in video tampering localisation.

Although video tampering detection techniques exist, these are often tailored to specific tampering techniques and evidence suggests that they do not always generalise. The authors of [17] produced FaceForensics, one of the largest manipulated video datasets to date, based exclusively on the digital re-enactment strategy Face2Face [6]. Authentic sources were taken from YouTube and used to perform digital puppetry [6]. A deep neural network was successfully trained to detect manipulated video content with less than 1% error where humans did little better than guessing, but in [80] is was shown that a similarly trained network completely failed to detect digital tampering in other sequences. The authors of [80] compiled their "Fake Face in the Wild" (FFW) dataset by mining content from YouTube which already contained digital facial manipulations. They also used SwapMe from [204], which is a dataset of swapped faces on JPEG images. The fake faces in both FFW and SwapMe went almost completely unnoticed by an Inception network trained on FaceForensics. This result implies that there is some consistency within FaceForensics which is absent in SwapMe and FFW. One likely source to be overlooked by human eyes is compression. SwapMe has JPEG compression, and FFW, like the FaceForensics authentic sources, is subject to

YouTube compression. The high quality FaceForensics manipulated content, however, is uncompressed, or, rather, lacks features associated with common YouTube levels of compression, as we demonstrate in this chapter.

Machine learning techniques are evidently very good at discovering consistencies and patterns within data and are successfully used to detect tampering on singular datasets[17, 2], but techniques are required to fulfil their large data requirements. In [168], a Siamese neural network was used to identify whether image patches exhibited consistent image metadata and could, therefore, have come from the same image pipeline and be part of the same authentic image. Their network was trained using only authentic images and their associated EXIF metadata, so a large dataset could be gathered quickly and simply. Using 80 features from EXIF metadata and 3 further processing techniques (JPEG compression level, Gaussian blur and re-scaling), the authors managed to classify whether two 128x128 pixel patches were consistent with each other or not and achieve a new state-of-the-art in image tampering localisation. Also in the image domain, similar methods were used in [204] on face swapped images: a large dataset of face swapped images was generated using two different methods of face swapping. A triplet network was used to ascertain whether given patches were more consistent with background patches from the same image or from different images. The second network of the two-stream architecture was trained to classify faces as authentic or fake. The network trained on SwapMe achieved accuracy of 99.5% and 82.9% when tested on SwapMe and FaceSwap test sets, respectively.

While image metadata is often available in online image files, authentic video metadata is not as readily available. Video files are much larger and will often be edited or compressed and recompressed for storage or streaming purposes. Recompression can also mask tampering, but some evidence can remain.

The method of tampering detection presented in [16] relied on the availability of several authentic frames. The authentic frames were used to train a neural network autoencoder, which “encoded” 128128 pixel patches to their quintessential components and then “decoded” them back to their original size. Because the autoencoder was trained using only pristine patches, spliced content resulted in a large, detectable error between the output of the autoencoder and the pixels of the actual patch. This error was used to classify frames as tampered or authentic. The method relied on the availability of authentic frames from the sequence under test, however, and these cannot be guaranteed.

7.2 A Framework for Video Tampering Detection

The full proposed framework for tampering localisation is summarised in Figure 7.1. First the sequence is split into 80x80 patches. This allows for differing frame dimensions and differing sequence lengths. Only intra frames are selected to overcome the limitations associated with the current trained networks, and these are identified using the methods described in Chapter 6. With a fully labelled dataset, individual patches can be labelled as tampered or untampered according to the mask. For simplicity, the minimum tampered region in a patch necessary to assign a "tampered" label is 1 pixel. The features are currently features of compression, and the CNNs used to extract them are trained using authentic data as in Chapters 5 and 6. The features used are Quantisation Parameter (QP), deblocking filter and inter/intra mode. The patch feature vectors can then be used to train a simpler classifier, and in this case we used Random Forest [205] with 100 estimators. We also tested Decision Trees, Naive Bayes and Support Vector Machine classifiers, but Random Forest gave the best results, although all four classifiers were very similar.

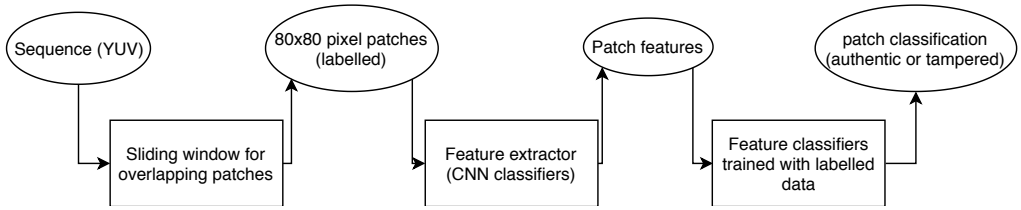


Figure 7.1: Proposed tampering detection framework

The use of 80x80 patches allows for tampering localisation within frames of a video sequence. It yields more samples from a smaller dataset, and allows both tampered and authentic patches to come from the same sequence. Tampered datasets can also be balanced if they are represented as a small number of patches. Class imbalance is an inherent challenge in tampering localisation with many sequences consisting of far more authentic pixels than tampered.

7.3 Datasets for Tampering Detection

As seen in Section 3.4, large and varied datasets for video tampering detection are in short supply. Three publicly available datasets were used for evaluation: FaceForensics [17], D'Avino et al [16] and Video Tampering Dataset (VTD) [15].

For D'Avino and VTD, "frame deltas" were calculated for 16x16 pixel patches to correspond with the QP prediction values as described in Section 6.2. Frame deltas were not calculated for FaceForensics, since only a single frame was used.

7.3.1 FaceForensics: Synthetic Regions

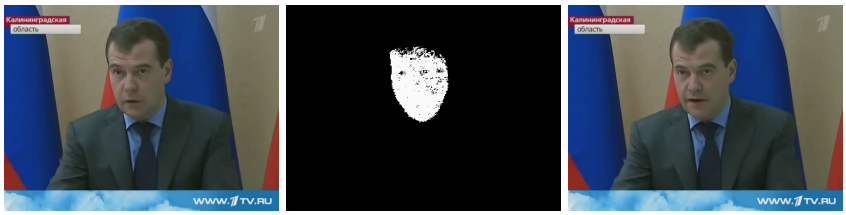
FaceForensics [17] is a large, tampered video dataset. The content is restricted to talking heads, including news readers, with minimum dimensions of 480p and 300 frames. The authentic source videos originally came from YouTube and, as such, began their life compressed. In this dataset, video manipulation is done using Face2Face [6], and, when present, tampering begins on the first frame of the sequence. Every tampered video has an authentic counterpart, and the video sequences are supplied as losslessly compressed files. The sequences are also available compressed, but we leave this dataset for future work.

Only the first frame of every sequence in the dataset was used and there is no guarantee that every first frame is an intra frame. In order to balance the dataset and speed up processing, crops of the tampered areas and corresponding authentic areas were created by using the difference between related authentic and tampered sequences. Areas outside of the crops were pixel-wise identical between tampered and authentic content. Figure 7.2 gives an example from FaceForensics and shows how cropping was done.

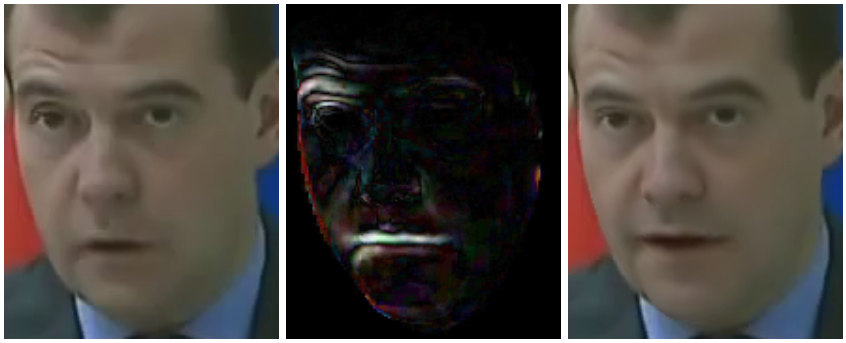
Crops of the first frame of each sequence were split into 80x80 patches. The individual patches were then passed through pre-trained CNNs from Chapters 5 and 6, to estimate their QP parameter along with inter/intra mode and deblocking setting. The patches, now represented by 3 values (QP val, intra/inter mode and deblocking filter setting) were then used to train and test several basic, supervised machine learning classifiers: Naive Bayes, Decision Tree, Random Forest and Support Vector Machine (SVM). The test/train splits defined in the dataset were used.

7.3.2 D'Avino's Spliced Video Dataset

The dataset provided by D'Avino et al [16] consists of 10 spliced videos. The sequences are all 720p and 281-488 frames in length. Each sequence is a single camera, continuous scene, although the camera is not always static and some sequences are subject to significant camera motion. Individual frames of the sequence do not always contain a tampered region. Unlike FaceForensics, in some sequences



(a) Authentic, full (b) Binary diff (c) Tampered, full



(d) Authentic, crop (e) Difference (f) Tampered, crop

Figure 7.2: Example from FaceForensics.

of D'Avino, the tampered region does not appear in the first frame of the sequence. This meant that for our detection framework, intra frames had to be extracted using the method in Chapter 6, reducing the size of the dataset. Only sequences containing spliced content were considered, and masks were used to label pixels patches as authentic or tampered.

The dataset itself provides uncompressed .avi video files for original, forged and binary mask for each sequence, however the source videos used to create splices have been compressed in the past and evidence of compression can be found in the pixels (see Section 7.4). Original background videos were filmed by the authors, but content for the chroma-keyed regions was obtained from YouTube and other sources[1]. Splicing was achieved using Adobe After Effects CC.

The authors have benchmarked this dataset using an autoencoder which was trained on 50 authentic frames and then used to process 100 test frames from each sequence. The error between the autoencoder-constructed frame and the actual frame was then thresholded on a sliding-window basis to obtain a "tampered" or "authentic" classification. Using this method, the authors were able to obtain an average true positive rate of over 0.9 for an average false positive rate of 0.1.

[1]Some spliced content of [16] came from https://www.hollywoodcamerawork.com/green-screen-plates.html

7.3.3 VTD: Compressed Video Tampering Dataset

VTD [15] comprises 26 forged sequences and their 26 authentic counterparts. There are also 7 authentic sequences available in the dataset. The tampered video files comprise 10 sequences of spatio-temporal copy-move, 6 inter-frame tampering (frame shuffling), and 10 spliced sequences. For our purposes only 10 copy-move and 10 spliced sequences were used. The sequences are between 420 and 480 frames in length and are all available in 720p, barring a single 420p sequence. Some sequences contain cut scenes and there is evidence of non-motion compensated resampling within the dataset, which implies that source videos were not pristine.

The dataset is distributed via a YouTube channel and, as such, is subject to recompression. The videos were downloaded from YouTube selecting the highest possible bitrate and frame dimensions, and the average bitrate was 1.7 Mbps, which equates to a compression rate of 0.06 bits per pixel (bpp). Re-compression itself makes mask extraction noisy, and tampering localisation particularly challenging, as outlined in Section 3.5. The lack of mask provision for this dataset also highlights the somewhat philosophical question of whether a pixel which remains unchanged between authentic and tampered sequences, yet forms part of a tampered object, is considered tampered or not. However, data from the compressed bitstream is also available, allowing accurate identification of key frames from the most recent (YouTube) compression. VTD is, as yet, unbenchmarked.

For VTD, masks were extracted using a thresholded difference between each frame of the forged and corresponding authentic sequences. Pixels with a difference higher than the threshold were labelled tampered, and those below labelled authentic. Thresholds in the range 0 to 64 were selected manually for each sequence. The mask pixels were then filtered temporally, using majority vote across 3 frames consecutive frames to remove erroneous compression noise. Finally, morphological operations were applied to each frame for further consolidation of the mask.

Again, intra frames were extracted using details from the downloaded bitstream. It was found that this dataset contained very few key frames, typically only 3 per sequence and a tampered region coinciding with a key frame was not guaranteed.

7.4 Compression Feature Distributions

In order to first show that predicted compression parameters can be used to locate tampering in video frames, we first examine the compression feature distributions of

Table 7.1: Predicted QP on authentic and tampered pixels in detected key frames

Sequences	Average QP (mask=0)	Average QP (mask=1)	Average absolute diff.
FaceForensics [17] (Face2Face)	26.81	11.27	15.54
D'Avino [16] (splice)	20.00	9.77	10.28
VTD [15] (splice)	31.55	26.46	5.08
VTD [15] (copy-move)	32.43	34.18	3.85

Table 7.2: Compression Features in D'Avino's Spliced Dataset

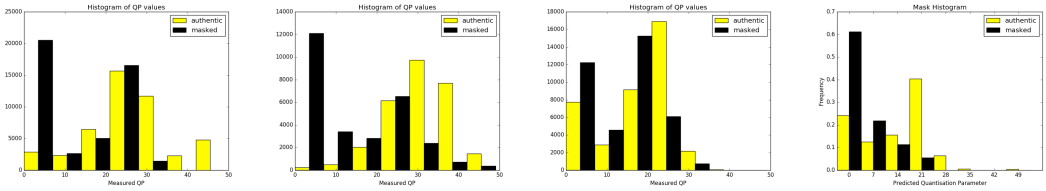
Sequences	QP (mask=0)	QP (mask=1)	Deblock (mask=0)	Deblock (mask=1)	I/P (mask=0)	I/P (mask=1)
01_TANK	22.13	12.26	0.64	0.62	0.70	0.80
02_MAN	27.85	14.95	0.69	0.90	0.61	0.84
03_CAT	12.38	7.80	0.85	0.75	0.90	0.96
04_HELICOPTER	17.83	10.13	0.34	0.76	0.94	0.96
05_HEN	23.31	10.36	0.59	0.91	0.85	0.89
06_LION	27.52	10.94	0.45	0.87	0.59	0.88
07_UFO	14.57	10.16	0.81	0.96	0.73	0.98
08_TREE	21.26	7.92	0.25	0.86	0.88	0.96
09_GIRL	17.60	6.71	0.51	0.86	0.77	0.86
10_DOG	16.03	7.07	0.61	0.87	0.70	0.91

the datasets. Table 7.1 shows predicted QP, averaged over the regions defined by the binary tampered mask. The last column in Table 7.1 shows the absolute difference in average QP averaged over all sequences for D'Avino and VTD. It is necessary to look at the difference in QP within individual files in this way because although the tampering method is consistent throughout the dataset, the *sources* differ. The hypothesis is that spliced patches will differ from authentic patches within a sequence, **not** that there will be a similarity between spliced content of a given dataset.

It can be seen in Table 7.1 that there is a distinct difference in QP averaged over authentic and tampered regions, particularly for the spliced content of [16] and the digital re-enactment content of [17] where the average absolute difference is larger than the granularity of the QP classifier.

Figure 7.3 shows the predicted QP class distribution for some sequences from [16]. Table 7.2 shows the averages per sequence. This shows that authentic regions and spliced regions display two different quantisation parameter distributions. The masked content in general has a lower QP while the authentic background content displays higher QP. The sequences of [16] consist of authentic content filmed on hand-held camera phones spliced with green screen plates. It can be deduced that, for these sequences, the hand-held cameras produced video of a lower quality than the green

screen plates, resulting in distinct differences in QP distribution. Similarly, the higher quality tampered data evidences more features consistent with the application of a deblocking filter. It cannot be ascertained whether this reflects the original settings of the source sequences, but it does allow a degree differentiation between tampered and authentic patches in some sequences. The inter/intra feature provides the least distinction between tampered and authentic patches, but the spliced patches, in this case, are more likely to be classified as “inter”. Given that this test was run over only intra frames, and that the intra frame decision was made using frame deltas and QP, this demonstrates that the I/P feature may not be particularly accurate on real data with constant bit rate, however it may still be useful for differentiating content in the same frame from two different sources.



(a) 05_HEN (b) 06_LION (c) 07_UFO (d) 08_TREE

Figure 7.3: Predicted QP class distribution for authentic and spliced content in detected key frames of tampered sequences from [16]

Figure 7.4 shows graphs of QP distribution of some sequences from VTD. Table 7.3 shows results for the splice and copy-move content. The copy-move content does not display a marked difference in predicted QP parameters because all copy-move content comes from within the same sequence and, hence same QP distribution. The sequence “dahua” has an overall average key frame QP of 34.36, with authentic content of 34.40 and tampered content average 33.93. The difference for spliced content is slightly higher, and the spliced sequence shown in Figure 7.4 has overall average key frame QP of 32.69, with authentic content of 33.14 (variance 15.25) and tampered content average 25.88 (variance 6.11). This is not significant enough for our CNN QP predictor to ascertain with high levels of accuracy which distribution individual regions come from. The training set for our QP predictor used QP steps of 7, and the difference between spliced and authentic content of VTD is smaller than this. This effect can be attributed to the re-compression step in the processing of this video: if the quality of both spliced and authentic content was reduced during re-compression, then any differences in QP distribution will be consequently smoothed. Frame shuffling may exhibit differences in QP, but our current technique is limited by its reliance on key frame detection.

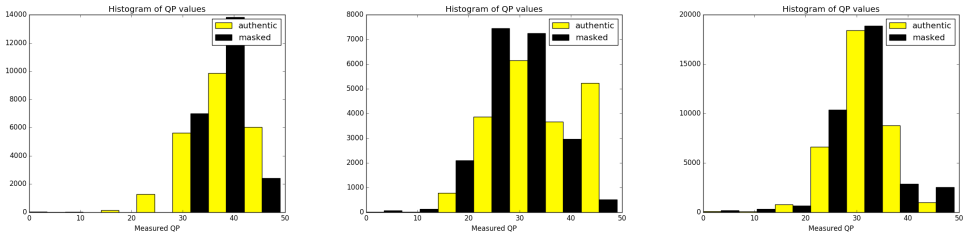
Table 7.4 shows frame deltas (Section 7.3) averaged over regions and sequences for

Table 7.3: Compression Features in VTD Dataset

Sequences	QP (mask=0)	QP (mask=1)	Deblock (mask=0)	Deblock (mask=1)	I/P (mask=0)	I/P (mask=1)
Spliced Sequences						
billards	28.72	25.01	0.37	0.55	0.77	0.86
studio	30	24.68	0.71	0.25	0.73	0.88
plane	35.17	33.12	0.31	0.31	0.73	0.97
bowling	32.37	25.16	0.35	0.22	0.71	0.97
kitchen	32.71	23.59	0.37	0.68	0.47	0.92
passport	26.64	23.29	0.73	0.88	0.64	0.63
highway	35.46	-	0.52	-	0.57	-
carpark	33.36	29.39	0.22	0.61	0.33	0.6
carplate	32.84	27.42	0.3	0.42	0.8	0.89
cake	32.11	26.46	0.58	0.6	0.49	0.68
Copy-move Sequences						
swimming	32.96	35.05	0.41	0.09	0.65	0.84
archery	31.81	28.3	0.47	0.22	0.6	1
basketball	33.83	39.08	0.55	0	0.68	1
camera	33.02	42.57	0.36	0.1	0.78	0.98
cctv	29.34	-	0.39	-	0.74	-
clarity	29.95	-	0.24	-	0.62	-
dahua	34.4	33.93	0.16	0.44	0.78	0.73
football	31.1	26.69	0.69	0.81	0.44	0.31
manstreet	30.01	34.98	0.33	0.17	0.44	0.34
whitecar	32.3	32.87	0.42	0.72	0.51	0.49

VTD and D'Avino. It can be seen that the averaged frame deltas for the tampered and authentic content of [16] are very close. There is a much bigger difference in VTD's spliced content. This is because some sequences, such as "Forgery Billiards" and "Forgery Studio" simply used static images as their spliced content. Similarly, some of the copy-move sequences, such as "Forgery basketball skills" and "Forgery 100m swimming", also used static content, however since the tampered areas are also relatively static, it is not clear if this is an explicit feature of the tampering itself or simply of the region that was tampered.

Figure 7.5 shows the distribution of QP parameter over tampered and authentic content of the FaceForensics dataset. Figure 7.5b shows the unmodified distribution over the complete first frames of the test set. It demonstrates the imbalance problem that is common in video tampering: there exists far more authentic content than labelled, tampered content, even in tampered datasets themselves. Figure 7.5c shows the same distributions but oversamples the tampered class to achieve a better balance. Figure 7.5a shows the distribution over the complete dataset when it is balanced by

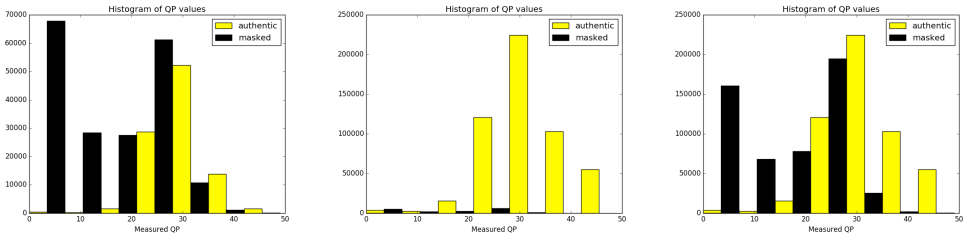


(a) Copy-move: dahua (b) Splice: bowling (c) Shuffle: swann

Figure 7.4: Predicted QP class distribution for authentic and spliced content in detected key frames of sequences from VTD

Table 7.4: Predicted frame deltas on authentic and tampered pixels in detected key frames

Sequences	Average Diff (mask=0)	Average Diff (mask=1)	Average absolute diff.
VTD [15] (copy-move)	0.68	0.43	0.31
VTD [15] (splice)	0.72	0.56	0.32
D'Avino [16] (splice)	0.90	0.92	0.11



(a) Cropped (b) First frame (c) Oversampled First Frame

Figure 7.5: The distribution of predicted QP in FaceForensics

cropping the masked regions from both the authentic and altered subsets. The QP parameter distribution for all sequences in the dataset can be plotted on a single graph because all the authentic sequences came from a single source (YouTube) and all the tampered sequences have been tampered using the same technique. In Figure 7.5a, it can be seen that while the authentic content has a mean value of QP=26.8, the tampered content has a much lower mean of 11.3. In Figure 7.5b (the test set), authentic data has QP=28.9 and the tampered content 12.2 This corresponds to QP=28 and QP=14 in our QP classifier. A T-test also showed that the tampered content QP distribution is distinct from the authentic content distribution. The distributions are also different shapes with the tampered content producing much higher response in the "uncompressed" class QP=0, and QP=7.

Table 7.5 shows the compression features for different subsets of the FaceForensics

Table 7.5: Compression Features in FaceForensics Data Subset

Set	QP (mask=0)	QP (mask=1)	Deblock (mask=0)	Deblock (mask=1)	I/P (mask=0)	I/P (mask=1)
Crop (train)	26.81	11.28	0.85	0.95	0.30	0.79
Crop (test)	26.71	10.54	0.84	0.97	0.31	0.83
Crop (val)	26.83	11.51	0.86	0.96	0.30	0.78
Full Frame (test)	28.92	12.20	0.66	0.95	0.50	0.81

Table 7.6: Ablation Study of Compression Features in Cropped FaceForensics Dataset

Features	TPR	TFR	Average
quant	0.94	0.68	0.81
deblock	0.96	0.15	0.56
i/p	0.80	0.70	0.75
quant, deblock	0.93	0.71	0.82
quant. i/p	0.86	0.87	0.87
i/p, deblock	0.77	0.76	0.77
quant, deblock, i/p	0.86	0.89	0.88

dataset. It can be seen that the cropped subsets (test, train and validation) have roughly similar averages for features. Using the full frame instead of a crop results in a slight rise in the QP of tampered regions, which could be attributed to our method of labelling some patches as tampered even when they contain only a very small tampered proportion. In the cropped dataset, these ambiguous patches have been mostly cropped out. The rise in QP in the authentic regions can be partly attributed to features of the background itself or encoder choices made during compression.

7.5 Tampered or Authentic?

The proposed framework from Section 7.2 was now put into effect. The differences in distribution in the cropped FaceForensics dataset allowed individual 80x80 patches in the test set to be classified as tampered or authentic with 81.46% using QP alone when analysed with Random Forest and SVM. This rose to 88.1% accuracy with all three compression features, giving an F1 score of 0.881. Cropping the dataset like this resulted in a completely balanced dataset, so accuracy and F1 score were identical. Results were slightly lower on the validation set with 86.3% accuracy. Using a simple majority voting system on the patches of FaceForensics, these techniques were able to correctly classify 96% of the cropped first frames from the test set as either authentic or tampered.

Table 7.7: Prediction Results for a Random Forest Classifier trained with Cropped FaceForensics Training Data. Final column illustrates dataset imbalance.

Test Set	TN	FP	FN	TP	TPR	TNR	% +ve
FaceForensics crop test	9562	1313	1271	9604	0.88	0.88	50
FaceForensics crop val	20892	2364	4010	19246	0.83	0.90	50
FaceForensics test	454868	70807	2629	13916	0.84	0.87	3.05
D'Avino all	191152	190930	2520	14398	0.85	0.50	4.24
D'Avino low	77798	137099	697	6406	0.90	0.36	3.20
D'Avino high	113354	53831	1823	7992	0.81	0.68	5.55
VTD splice	247438	29406	4189	967	0.17	0.89	1.83
VTD copy move	294358	14503	3118	21	0.01	0.95	1.01

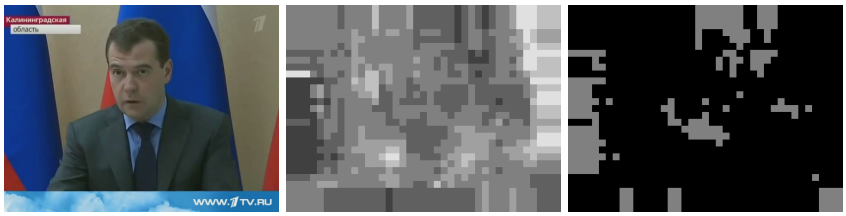
An ablation study of the features themselves was performed by training a random forest on each of the three compression features individually and in their various combinations. Only the cropped FaceForensics datasets were used. The results, Table 7.6, demonstrate that the use of all three features gives the best balance between true positives and true negatives. The quantisation parameter gave the best individual feature result but it was more prone to false positives. Adding both deblock and i/p features rectified this.

Applying a random forest trained on cropped FaceForensics training patches to the full first frame patches of the FaceForensics test set resulted in many false positives. Figure 7.6 shows an example. It can be seen that the QP is particularly low, classified as uncompressed, centred on the tampered region of the face. Other areas in the background of the frame, however, also have relatively low quantisation and this causes them to be classified as tampered.

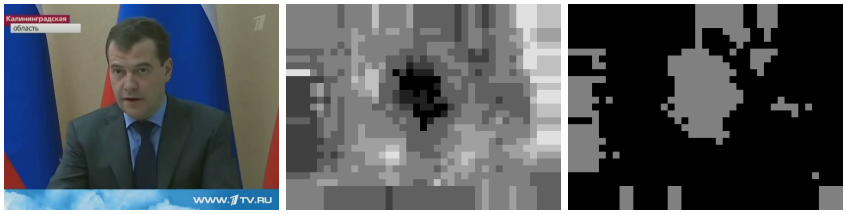
Table 7.7 shows the results of this random forest classifier on all datasets in terms of True Negatives (TN), False Positives (FP), False Negatives (FN), True Positives (TP), True Positive Rate (TPR), True Negative Rate (TNR) and the percentage of all samples that were labelled as the positive (or tampered) class. For the VTD and D'Avino datasets, full intra frames were extracted using the method in Chapter 6. The FaceForensics datasets were either the cropped (and therefore balanced) datasets or the first frame of each sequence for the test set only. The full first frame FaceForensics dataset exhibits a large dataset imbalance because the tampered areas occupy relatively few pixels compared to the background pixels (which are identical between matched tampered and authentic frames). Approximately only 3% of all samples in the dataset are tampered. This leads to poor F1 scores due to a high number of

(a) Authentic (b) Authentic, QP (c) Authentic, Class

(d) Tampered (e) Tampered, QP (f) Tampered, Class



(g) Authentic (h) Authentic, QP (i) Authentic, Class



(j) Tampered (k) Tampered, QP (l) Tampered, Class

(m) Authentic (n) Authentic, QP (o) Authentic, Class

(p) Tampered (q) Tampered, QP (r) Tampered, Class

Figure 7.6: Full frame results from FaceForensics.

false positives relative to true positives. TPR and TNR, however, are unaffected by the large imbalance and therefore remain consistent between cropped and full frame datasets. This shows that features of compression are useful for tampering detection and localisation.

Table 7.7 also shows that dataset imbalance is also present in D'Avino's dataset and VTD. In D'Avino, 4.24% of patches are tampered, but this can vary between sequences. For the spliced content of VTD, less than 2% of patches are tampered and in the copy-move sequences of VTD, only just over 1% of patches are tampered. The classification results, however, show that a random forest trained on the balanced, cropped frame patches of FaceForensics can still identify the majority of tampered patches in D'Avino. This is because the compression feature distribution of D'Avino's tampered and authentic patches is quite similar to that of FaceForensics. The high number of false positives can be attributed to D'Avino's sequences having lower QP in general than FaceForensics, and half of the sequences have QP much lower than that in FaceForensics, as can be seen in Table 7.2. If D'Avino is split into two sets for high and low authentic QP, the set containing high QP exhibits fewer false positives. This yields results that are almost comparable to D'Avino's benchmark with in terms of true positive rate of 0.85 (compared with 0.9). Although the benchmark false positive rate is much lower than ours, it should be emphasised that this classifier was trained on a different dataset entirely and does not require separate identification of pristine frames in the sequence. The same random forest classifier performs very poorly on VTD. It cannot detect the copy-move patches because these patches come from within the same sequence. It also fails to detect the majority of spliced patches, because it relies on absolute QP value. The VTD sequences have been recompressed by YouTube, so very few patches occupy QP=0 and QP=7 classes, which signify tampered content in both FaceForensics and D'Avino, and those that do may even be incorrect classifications.

The results on FaceForensics demonstrate how tampered content, no matter how realistic it appears to human eyes, does not necessarily emulate compression patterns that can be found in authentic data. Moreover, it may be that the higher quality of the tampered content of FaceForensics contributed to the inability of human viewers to differentiate between tampered and authentic content. Human viewers may naturally prefer high quality content and be more likely to declare it authentic. This result also helps to explain the findings of [80], where it was demonstrated that a CNN trained on a subset of FaceForensics did not generalise at all well to the authors' own dataset "Fake Faces in the Wild" (FFW). The FFW dataset comprises a list of YouTube handles

along with start/end times. Sequence-wise, the content is entirely non-authentic, however manipulation techniques vary from CGI regions, to face swapping and the use of FakeApp. Because all the frames in FFW come from YouTube, they are ALL subject to YouTube compression or recompression, similar to VTD. All the manipulated frames in uncompressed FaceForensics are subject to the original YouTube compression everywhere **except** in the tampered region. If the CNN trained in [80] relied on features that specifically related to uncompressed faces to specify tampering, then it would fail to detect fake faces which are compressed to the same levels as the authentic content. As shown in Chapter 4, CNNs trained on high quality data are less robust against increasing levels of compression. The network in [80] effectively used almost uncompressed tampered pixels for training, but was applied to much lower quality data. This alone would cause a drop in accuracy, even assuming that other tampering features were consistent between FFW and FaceForensics.

7.5.1 Unsupervised Clustering

Tampering techniques vary from sequence to sequence, and may vary within datasets in future. Classifiers trained on a particular dataset will always be subject to the distribution of that dataset and may not generalise to other sequences which use different tampering methods. A more generalisable method of differentiating between tampered and authentic patches would use only patches from within the suspicious sequence itself. Unsupervised clustering is one possibility to achieve this, although clustering is not classification. Clustering will simply group like samples together and not necessarily label the classes correctly. Because of this, we use Matthews Correlation Coefficient (as described in Section 2.2.3) for evaluation.

Using all three compression parameters from the cropped FaceForensics test patches, and unsupervised k-means clustering, assuming two clusters, we achieved MCC of 0.67 and mean F1 score of 0.81. Clustering could be validly performed over the whole dataset because the tampering method is the same for all sequences, and the source video footage all came from a single platform. Unfortunately k-means clustering did not perform as well with full first frames of FaceForensics, achieving MCC=0.11, but when this dataset was balanced by random oversampling, MCC rose to 0.58 with mean F1 over both classes of 0.76.

Using only frame deltas and predicted QP as features and forming two clusters using unsupervised k-means clustering, the following were achieved on D'Avino et al's dataset [16]: mean MCC: 0.249, mean F1 score: 0.255, mean IoU: 0.163. MCC rises to 0.302 if the two lowest scoring sequences ("07_UFO" and "03_CAT") are ignored.

In both these sequences, spliced objects occupy relatively few key frames, and the tampered region in key frames is particularly small. Moreover, QP distributions for authentic and tampered regions in these two sequences are much less distinct than in the other sequences, and fall under the QP step size of 7 necessary for our CNN QP predictor.

In all the sequences of [16], key frames were estimated to occur at an interval of one every 30 frames. While this dataset is supplied as uncompressed, it is evidently compiled from compressed sources and one key frame per second at a frame rate of 30 fps is relatively standard in compression. Figure 7.7 shows example frames for some of the sequences from [16]. It can be seen that the quantisation parameters in tampered regions is generally lower than in authentic content. Unsupervised clustering of predicted QP and frame deltas was used to group pixel patches into one of two groups. These groups were nominatively labelled “authentic” or “tampered”, however given that some tampered video content is simply two or more authentic videos spliced together, these labels could be effectively switched to more closely match the ground truth.

Figure 7.7: Heatmap for test sequence (top to bottom) 08_TREE, 05_HEN and 06_LION from [16]: (left to right) real, fake, ground truth, clustered data, QP predictions. Darker areas mean lower QP predictions

The results on VTD [15] were somewhat less encouraging. While key frames were detected every 30 frames in [16], the most common gap between key frames in VTD was 160 frames, resulting in only 3 key frames for over half of the sequences in the dataset. Some sequences completely lacked key frames coinciding with any tampered regions, and these were removed from our analysis. Excluding these sequences, the mean MCC was 0.082, F1 0.065 and IoU 0.035. This shows little correlation between predicted and actual tampered areas, which can be partly attributed to re-compression causing an equalisation in the QP distribution between tampered and authentic regions. Although a realistic process, re-compression of this dataset also

resulted in challenges in extracting an accurate tampering mask and this may also be a contributory factor. The authors of [16] also noted that YouTube compression had a negative effect on their autoencoder-based tampering detector. This highlights challenges for tampering detection in video distributed using one of the most common video-sharing platforms in the world. Further work is needed if tampering detectors are to thoroughly overcome the challenges of re-compression.

7.6 Conclusions

With video manipulation techniques currently increasing at an unprecedented rate, it is vital to develop features that can detect tampering irrespective of the original tampering method. A lack of large, current, comprehensive tampered video datasets and the huge imbalance in existing tampered datasets makes learning these features directly from tampered data impossible. Therefore it is necessary to derive such features using mainly authentic sources. Video compression provides a common foundation for video analysis, with the vast majority of available video sequences compressed in some format. Moreover, the use of machine learning techniques and feature discovery from data provide a methodology which can be used to produce updated features should new compression standards fall in to common use.

We have shown that three features of H.264/AVC compression, namely quantisation parameter, intra/inter mode and deblock mode, can be used to aid localisation of tampered regions within key frames. Results suggest that this type of feature shows great promise in the work towards universal tampering detection. Video manipulation causes self inconsistencies within the video sequence, whether this is caused by splicing, inpainting, inter-frame tampering or small, localised changes used to alter content such as those used in digital re-enactment. These inconsistencies can be made visible using different data representations. We have shown that with the use of only four features (QP, inter/intra, deblocking and frame differences) derived only from untampered sources, self inconsistencies within a video sequence can be detected and exploited to localise tampering.

Also notable is the large dataset imbalance present in tampered datasets which may become a large problem in tampering localisation. The results on the FaceForensic cropped and first full frame datasets also suggest that k-means is not the best method to perform unsupervised clustering on such imbalanced datasets. Alternatives might be to use deep metric learners, such as Siamese neural networks as used in [168] for image manipulation detection, however we leave this area for future work.

Chapter 8

Conclusions

This chapter summarises the findings of the preceding chapters and discusses limitations evident within the work.

In this work we have shown how:

- Video manipulation techniques are increasing in both number and realism [60]. Techniques which completely fool humans are already available, and although there already exist DNN-based methods that can reliable distinguish between authentic and tampered video, these are not always generalisable between different tampering techniques.
- Video compression has a negative effect on classification in DNNs [24], with higher compression levels causing lower accuracy in classification. The selection of appropriately compressed training datasets can go some way to improving classification accuracy, however only if the compression level of the sample under test can be ascertained.
- Compression parameters can be determined directly from pixels in video sequences [25], subject to some limitations.
- Compression parameters derived from pixels can be used to localise tampered regions in video sequences [26]. The explainability of these features allows them to be generalised to different datasets.

This thesis has provided a thorough review of the current trends in video manipulation (Chapter 3). In the last few years, more and more manipulation techniques have been presented, and many of them no longer fit in with traditional manipulation categories. Moreover, there is evidence that these manipulations are rapidly developing to

become undetectable for human eyes. In the very near future, humans viewers may be completely unable to identify video that has been tampered, let alone identify the method of tampering.

Although methods of tampering detection exist and some of them are very effective, many of them focus on a single type of manipulation. This can be partly attributed to the lack of large and varied tampered video datasets. While many tampered image datasets already exist, tampered video datasets still lag behind in number and quality. They suffer from non-standardised distribution processes which cause problems in their use. We have highlighted some problems, particularly those related to compression, which can inadvertently cause tampered video datasets to be much more difficult than the benchmarked original. Yet it remains vital to develop tampering detection techniques which can handle multiple different types of tampering. If human viewers are incapable of ascertaining whether tampering is present or not, then they cannot be expected to differentiate between different types of tampering. Moreover, since new digital video manipulation methods are being developed at an unprecedented rate, specific detection methods will be required at an equivalent rate simply to keep up. It is far more efficient to develop generalisable methods related to video forensics.

In the chapters presented here, we have seen how the effects of video compression on pixels can be used as an ally rather than an enemy. We have examined how compression affects the performance in deep neural network classifiers and shown how these problems can be observed in the features of the very first layer in the network (Chapter 4). Although low quality video will always cause a reduction in observed classification accuracy, some of this reduction could be offset by applying the same level of compression to the training data. This is an important find as it gives insight into how transferable features can be developed.

We have also presented a novel method to derive specific compression features directly from the pixels of a video sequence (Chapter 5). Using standard layers, a CNN can extract compression features such as quantisation parameter, intra/inter mode and deblocking filter settings from a relatively small image patch. Accuracy on these features is sufficient that these can be used to uncover evidence of the processing history (Chapter 6), and in some cases, identify frames that were previously compressed using a different mode.

We have shown how compression-related features show great promise in the field of tampering detection (Chapter 7). Most importantly these are features which can be

learned from authentic or synthesised data, thus satisfying deep learning's data hungry needs without a requirement for a perpetually updated, large and varied tampered video dataset. The use of synthesised datasets also provides a convenient means to verify the authenticity of the training set. With some manipulated video already available online and some tampering techniques already in the public domain, it is only a matter of time before datasets scraped from publicly available sources are subject to a degree of pollution by tampered video. Some deep learning video manipulation detection techniques are trained using very specific datasets on which they achieve high accuracy, and we have shown how some of this accuracy may be attributable to compression features. Deep neural networks will learn whichever features yield the best results. These are not always explainable, but explanation yields enlightenment. Specifically, if video tampering detection methods depend on compression features, then disruption of these features via recompression of the sequence will obviously disrupt the detector itself. The ability to explain how these deep neural networks function allows for better predictions to be made about their transferability. By training neural networks to identify known, multi-variable features, we can enhance explainability.

8.1 Limitations and Future Work

The work in this thesis has shown that video compression features can be useful in the detection of video manipulation, and with that in mind, there are a number of improvements that can be made.

8.1.1 More Accurate Compression Features

The novel methods of compression parameter detection presented here, although accurate to the extent that the features are useful, could be made more accurate still. With a better level of accuracy, more data could be gleaned from the pixels, thus improving the overall method in tampering detection. This may involve more specialised architectures or novel neural network layers, or it could involve a shift to working in the frequency domain. Results in [165] showed that combining spatial and frequency domains was effective in image forensic analysis, and the video frequency domain has been utilised already in [121].

In the experiments in Chapter 5, accuracy was particularly low for predicted frames and this is a problem since the majority of compressed data is predicted frames. While we have worked around this problem in this first instance by creating a method to

identify key frames, a better solution would be to create techniques to accurately derive quantisation parameters from predicted frame patches. Indeed, this may be necessary as some video encoders use periodic intra refresh, and refresh frame regions rather than risking bitrate spikes by using key frames.

A complete analysis of a compressed bitstream might allow pixel residuals to be used to improve the accuracy of compression feature estimation for predicted frames. This would broaden the applicability of these methods to complete sequences, and help to overcome the shortage of key frames in modern compression implementations. In the ideal situation, the pixel patch size would also be minimised so that video tampering could be localised to a very accurate region. Although 80x80 pixels is good enough to see some success in tampering detection, a smaller patch size would be an improvement.

8.1.2 Features of Recompression

Although compression features have shown their effectiveness in video tampering detection, it can be anticipated that recompression still provides one of the main challenges. Recompression is the simplest way to equalise the histograms of authentic and tampered content presented in Chapter 7 and effectively conceal any evidence of tampering. To overcome this limitation, it is most practical to simply directly use recompression itself as a feature.

Further work is needed to ascertain the extent to which recompression detection can be learned from authentic sources and synthesised datasets. From the work here, it can be seen that some synthesised video content displays compression parameters which are similar to uncompressed content. If a distinction can be made between video patches that are compressed only once and those that are multiply compressed, then synthesised regions can still be distinguished from authentic regions even through recompression of the complete sequence. Ultimately, tampered video patches will have undergone one fewer compression than their authentic counterparts from the same sequence. If the number of recompressions can be ascertained for specific image patches, then this would effectively overcome the problem of recompressing tampered content to conceal manipulated areas.

Features of recompression may already be implicit in some DNNs. Work in [160] showed that the majority of tampered images of CASIA [44] exhibited double compression while authentic images were singly compressed implies that the exceptionally high tampered/authentic classification accuracy (97%) in [159] may have been

attributable to recompression detection. The work in [165] demonstrated how image recompression detection in isolation is also particularly accurate. Rebroadcast image detection [2] by DNN was also particularly effective, and although the authors did not mention it, recompression is very likely to be part of the rebroadcast process, regardless of the method of rebroadcasting. When taken together, all of these results suggest that ordinary DNNs readily learn features related to image recompression, and these are particularly useful for image forensics and tampering detection. The results in [121] show that recompression can also be detected in videos. There is good reason to believe that recompression detection, and particularly localised recompression detection, will be an important area in video forensics and tampering localisation.

Combination of techniques from video forensics and video tampering detection may lead to improved results. Focussing on video forensic features allows detectors to become independent of tampering methods and allows the development of generic tampering-type-agnostic detectors. This is a very important research direction as video manipulation techniques become invisible to humans.

8.1.3 One-shot, Type-agnostic Tampering Detection

The ability to ascertain tampering using only the pixels from the sequence under test is an important research direction. There is no guarantee that any future manipulation technique will have an established detector associated with it, nor yet even a convenient dataset demonstrating this particular type of tampering. Furthermore, although video sequences themselves provide an abundance of data in the form of frames, individual frames can no longer be reliably manually labelled as authentic or tampered. The key to type-agnostic tampering detection and localisation in many cases is finding localised self-inconsistencies within individual sequences.

In the first instance, these methods may be based on clustering techniques where patches from a video under test are simply clustered according to their features and a spatially localised cluster is indicative of region tampering. This is similar to what was presented in Section 7.5.1. In future work, unsupervised clustering could be replaced with deep metric learners such as Siamese neural networks, as used in [168] for image manipulation detection.

Any new methods for video tampering localisation must also take into account the likely presence of class imbalance. Tampered videos may contain many more authentic pixels than manipulated, and analysis of the datasets used in Chapter 7 shows that this very common. Class imbalance is a well studied problem in the literature both in

terms of handling it and evaluating results.

8.1.4 Synthetic Video Detection

The detection of fully synthetic video may evolve to become a research direction in its own right, however at this moment in time, most AI-generated photo-realistic synthetic video content is not suitably convincing to human eyes. As noted in Section 3.2.5, one of the main challenges in synthetic video generation is evaluation techniques. In particular, Table 3.1 shows that many methods rely on user comparison with previous methods rather than user comparison with reality, and this itself is indicative of how convincing synthetic video is. Human eyes can still easily identify a large proportion of machine produced synthetic video, and truly convincing synthetic video remains largely in the hands of professional movie makers.

Photo-realistic, convincing, synthetic images, however, are currently a reality, and it will not be long before this research trickles into the video domain. When this happens, suitable methods to identify synthetic videos will be needed and these could also be applied to localise synthetic regions in tampered video. Some recent methods, such as [206, 207] have shown that GAN-generated images exhibit pixel value distributions that are much smoother and quite distinct from distributions of authentic content. This may corroborate our observations that synthetic pixel regions generated by [6] exhibited features similar to uncompressed content. Furthermore, Chapter 4 showed that classifiers trained on uncompressed images achieved the highest accuracy overall. This may imply that GANs are predisposed to produce high quality, uncompressed images, rather than mimicking the compression levels of their training sets. A full investigation into how this can be useful for synthetic image detection is left for future work.

8.1.5 Changing Compression Standards

It must also be recognised that video compression industrial standards are still evolving, too. What worked in the past for one standard may not work well for other standards. This has already been seen as MPEG-2 [28] was superceded by H.264/AVC [27]. Although H.264/AVC is currently a very popular video compression standard, it may not always be the case. The next standard in the series H.265/HEVC [41] has already been released, with methods that overcome compression artifacts evident in H.264/AVC such as banding [208]. Various factors, such as patent issues and hardware limitations, may slow or halt the advance of HEVC specifically, but there are other

compression standards available and research into efficient video compression methods continues apace. Video compression standards all seek to improve video quality and reduce bitrate.

Reduction in compression artifacts and changing compression structures ultimately limits transferability and makes some methods obsolete. Although the individual trained networks presented within this thesis may not transfer directly to future standards, the concepts and processes detailed herein still stand. DNNs are adaptable and should be able to exploit the compression artifacts of future industrial standards even if these become increasingly invisible to human eyes. In the future, the techniques presented here can be applied to different video compression standards in order to keep pace with the digital world.

8.1.6 Network Architecture Optimisation

New network architectures and associated hyperparameters for specific tasks are under constant development. Although the methods in this thesis can be considered "good enough" to obtain convincing results, we leave the discovery of optimal architectures for this particular task to future work. In Chapter 4, we used relatively shallow network architectures. Further work is necessary to ascertain whether the negative effects of compression on classificaiton are fully present in deeper networks. It may be that all the networks for different levels of compression can be efficiently combined in a single network, given enough network layers and parameters. Evidence in the literature implies that different network architectures may compensate for some compression. For example, in [17], the Xception network [122] achieved much better performance on compressed data than other networks specifically designed for tampered video detection tasks. Xception uses depthwise separable convolutions, which separately convolve over different channels in each layer, and this may allow for development of more specialised features.

8.2 Conclusion

Video compression is largely overlooked by researchers, or seen as a challenge rather than an opportunity. Instead, with some understanding, it can be a means to explain some of the inner workings of deep neural networks, and develop transferable techniques for tampering detection using authentic sources. We have demonstrated, through the contributions in this thesis, that it can give valuable insight into how DNNs

work. Compression features can highlight invisible weaknesses and reveal forensic fingerprints in both synthetic and authentic content. In a world where synthetic or manipulated video can be passed off as real, it is important to develop tools which will help us to distinguish between fact and outright fiction. The use of video compression features can even give machines a significant advantage over human eyes when computer vision looks *beyond the pixels*.

Bibliography

[1] Qadir G, Yahaya S, Ho AT. Surrey university library for forensic analysis (SULFA) of video content. In: IET Conference on Image Processing (IPR). IET; 2012. p. 121–126.

[2] Agarwal S, Fan W, Farid H. A diverse large-scale dataset for evaluating rebroadcast attacks. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2018. p. 1997–2001.

[3] Redmon J, Divvala SK, Girshick RB, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. CoRR. 2015;abs/1506.02640.

[4] Wu Y, Lim J, Yang MH. Online object tracking: A benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2013. p. 2411–2418.

[5] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L. Large-scale video classification with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2014. p. 1725–1732.

[6] Thies J, Zollhöfer M, Stamminger M, Theobalt C, Nießner M. Face2face: Real-time face capture and reenactment of rgb videos. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2016. p. 2387–2395.

[7] Chan C, Ginosar S, Zhou T, Efros AA. Everybody Dance Now. arXiv preprint arXiv:180807371. 2018;.

[8] Jiang H, Sun D, Jampani V, Yang MH, Learned-Miller E, Kautz J. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 9000–9008.

[9] Liao R, Tao X, Li R, Ma Z, Jia J. Video super-resolution via deep draft-ensemble learning. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 531–539.

[10] Jin M, Meishvili G, Favaro P. Learning to extract a video sequence from a single motion-blurred image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 6334–6342.

[11] Sajjadi MS, Vemulapalli R, Brown M. Frame-recurrent video super-resolution. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 6626–6634.

[12] Lowe DG. Distinctive image features from scale-invariant keypoints. International journal of computer vision. 2004;60(2):91–110.

[13] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems; 2012. p. 1097–1105.

[14] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2009. p. 248–255.

[15] Al-Sanjary OI, Ahmed AA, Sulong G. Development of a video tampering dataset for forensic investigation. Forensic science international. 2016;266:565–572.

[16] D'Avino D, Cozzolino D, Poggi G, Verdoliva L. Autoencoder with recurrent neural networks for video forgery detection. Electronic Imaging. 2017;2017(7):92–99.

[17] Rössler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Nießner M. FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces. arXiv preprint arXiv:180309179. 2018;.

[18] Bayar B, Stamm MC. A deep learning approach to universal image manipulation detection using a new convolutional layer. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security. ACM; 2016. p. 5–10.

[19] Dodge S, Karam L. Understanding how image quality affects deep neural networks. In: Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on. IEEE; 2016. p. 1–6.

[20] Shullani D, Fontani M, Iuliani M, Al Shaya O, Piva A. VISION: a video and image dataset for source identification. EURASIP Journal on Information Security. 2017;2017(1):15.

[21] Sitara K, Mehtre BM. Digital video tampering detection: An overview of passive techniques. Digital Investigation. 2016;18:8–22.

[22] Rössler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Nießner M. FaceForensics++: Learning to Detect Manipulated Facial Images. arXiv preprint arXiv:190108971. 2019;.

[23] Mandelli S, Bestagini P, Tubaro S, Cozzolino D, Verdoliva L. Blind Detection and Localization of Video Temporal Splicing Exploiting Sensor-Based Footprints. In: 2018 26th European Signal Processing Conference (EUSIPCO). IEEE; 2018. p. 1362–1366.

[24] Johnston P, Elyan E, Jayne C. Spatial effects of video compression on classification in convolutional neural networks. In: Neural Networks (IJCNN), 2018 International Joint Conference on. IEEE; 2018. p. 1370–1377.

[25] Johnston P, Elyan E, Jayne C. Toward Video Tampering Exposure: Inferring Compression Parameters from Pixels. In: Pimenidis E, Jayne C, editors. Engineering Applications of Neural Networks. Springer International Publishing; 2018. p. 44–57.

[26] Johnston P, Elyan E, Jayne C. Video Tampering Localisation Using Features Learned from Authentic Content. Neural Computing and Applications. 2019;.

[27] ITU-T. H.264 Advanced video coding for generic audiovisual services. ITU-T; 2016.

[28] ITU-T. H.262 Information technology - Generic coding of moving pictures and associated audio information: Video. ITU-T; 2012.

[29] ITU-T. H.263 Video coding for low bit rate communication. ITU-T; 2005.

[30] x264 team. x264, version core:148 r2762 90a61ec; 2017. Available from: http://www.videolan.org/developers/x264.html.

[31] Richardson IE. The H. 264 advanced video compression standard. John Wiley & Sons; 2011.

[32] Jasinschi RS, Veen TN, et al. Motion estimation methods for video compressiona review. Journal of the Franklin Institute. 1998;335(8):1411–1441.

[33] Zhu S, Ma KK. A new diamond search algorithm for fast block-matching motion estimation. IEEE transactions on Image Processing. 2000;9(2):287–290.

[34] Zhang Y, Zhang C, Fan R. Fast Motion Estimation in HEVC Inter Coding: An Overview of Recent Advances. In: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). IEEE; 2018. p. 49–56.

[35] Gregory RL. Eye and Brain: The Psychology of Seeing. McGraw-Hill paperbacks. McGraw-Hill; 1978. Available from: https://books.google.co.uk/books?id=0ZNqAAAAMAAJ.

[36] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing. 2004;13(4):600–612.

[37] Video Test Media [derf's collection];. Available from: https://media.xiph.org/video/derf/.

[38] Van den Branden Lambrecht CJ. Vision models and applications to image and video processing. Springer Science & Business Media; 2013.

[39] Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A. Overview of the H. 264/AVC video coding standard. IEEE Transactions on circuits and systems for video technology. 2003;13(7):560–576.

[40] Kristan M, Leonardis A, Matas J, Felsberg M, Pflugfelder R, Čehovin L, et al. The Visual Object Tracking VOT2016 Challenge Results. In: Hua G, Jégou H, editors. Computer Vision – ECCV 2016 Workshops. Cham: Springer International Publishing; 2016. p. 777–823.

[41] ITU-T. H.265 High efficiency video coding. ITU-T; 2016.

[42] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: Common objects in context. In: European Conference on Computer Vision (ECCV). Springer; 2014. p. 740–755.

[43] Abu-El-Haija S, Kothari N, Lee J, Natsev P, Toderici G, Varadarajan B, et al. Youtube-8m: A large-scale video classification benchmark. arXiv preprint arXiv:160908675. 2016;.

[44] Credits for the use of the CASIA Image Tempering Detection Evaluation Database (CASIA TIDE) V2.0 are given to the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Corel Image Database and the photographers. http://forensics.idealtest.org;.

[45] Korus P, Huang J. Evaluation of random field models in multi-modal unsupervised tampering localization. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE; 2016. p. 1–6.

[46] LeCun Y, Bottou L, Bengio Y, Haffner P, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278–2324.

[47] LeCun Y, Jackel L, Bottou L, Cortes C, Denker JS, Drucker H, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. Neural networks: the statistical mechanics perspective. 1995;261:276.

[48] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint. 2014;arXiv:1409.1556.

[49] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 1–9.

[50] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 770–778.

[51] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 1026–1034.

[52] Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization. arXiv preprint. 2016;arXiv:1611.03530.

[53] Geirhos R, Rubisch P, Michaelis C, Bethge M, Wichmann FA, Brendel W. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:181112231. 2018;.

[54] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV). Springer; 2014. p. 818–833.

[55] Bondi L, Lameri S, Güera D, Bestagini P, Delp EJ, Tubaro S. Tampering Detection and Localization through Clustering of Camera-Based CNN Features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2017. p. 1855–1864.

[56] Bondi L, Baroffio L, Güera D, Bestagini P, Delp EJ, Tubaro S. First steps toward camera model identification with convolutional neural networks. IEEE Signal Processing Letters. 2017;24(3):259–263.

[57] Diallo B, Urruty T, Bourdon P, Fernandez-Maloigne C. Improving Robustness of Image Tampering Detection for Compression. In: International Conference on Multimedia Modeling. Springer; 2019. p. 387–398.

[58] Torralba A, Efros AA. Unbiased look at dataset bias. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2011. p. 1521–1528.

[59] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision. 2015;115(3):211–252.

[60] Johnston P, Elyan E. A review of digital video tampering: From simple editing to full synthesis. Digital Investigation. 2019;.

[61] Le TT, Almansa A, Gousseau Y, Masnou S. Motion-consistent video inpainting. In: 2017 IEEE International Conference on Image Processing (ICIP). IEEE; 2017. p. 2094–2098.

[62] Wang TC, Liu MY, Zhu JY, Liu G, Tao A, Kautz J, et al. Video-to-Video Synthesis. In: Advances in Neural Information Processing Systems; 2018. p. 1144–1156.

[63] Bansal A, Ma S, Ramanan D, Sheikh Y. Recycle-GAN: Unsupervised Video Retargeting. In: European Conference on Computer Vision (ECCV). Springer; 2018. p. 122–138.

[64] Suwajanakorn S, Seitz SM, Kemelmacher-Shlizerman I. Synthesizing obama: learning lip sync from audio. ACM Transactions on Graphics (TOG). 2017;36(4):95.

[65] Liu MY, Breuel T, Kautz J. Unsupervised image-to-image translation networks. In: Advances in Neural Information Processing Systems; 2017. p. 700–708.

[66] Dong H, Neekhara P, Wu C, Guo Y. Unsupervised image-to-image translation with generative adversarial networks. arXiv preprint arXiv:170102676. 2017;.

[67] Karras T, Aila T, Laine S, Herva A, Lehtinen J. Audio-driven facial animation by joint end-to-end learning of pose and emotion. ACM Transactions on Graphics (TOG). 2017;36(4):94.

[68] Chen L, Li Z, K Maddox R, Duan Z, Xu C. Lip movements generation at a glance. In: European Conference on Computer Vision (ECCV); 2018. p. 520–535.

[69] Liu Z, Luo P, Wang X, Tang X. Deep learning face attributes in the wild. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 3730–3738.

[70] Singh RD, Aggarwal N. Video content authentication techniques: a comprehensive survey. Multimedia Systems. 2017;p. 1–30.

[71] Milani S, Fontani M, Bestagini P, Barni M, Piva A, Tagliasacchi M, et al. An overview on video forensics. APSIPA Transactions on Signal and Information Processing. 2012;1.

[72] Pandey RC, Singh SK, Shukla KK. Passive forensics in image and video using noise features: A review. Digital Investigation. 2016;19:1–28.

[73] Al-Sanjary OI, Sulong G. Detection of video forgery: A review of literature. Journal of Theoretical & Applied Information Technology. 2015;74(2).

[74] Khodabakhsh A, Busch C, Ramachandra R. A taxonomy of audiovisual fake multimedia content creation technology. In: 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE; 2018. p. 372–377.

[75] Wu Y, Jiang X, Sun T, Wang W. Exposing video inter-frame forgery based on velocity field consistency. In: Acoustics, speech and signal processing (ICASSP), 2014 IEEE International Conference on. IEEE; 2014. p. 2674–2678.

[76] Sitara K, Mehtre B. A comprehensive approach for exposing inter-frame video forgeries. In: Signal Processing & its Applications (CSPA), 2017 IEEE 13th International Colloquium on. IEEE; 2017. p. 73–78.

[77] Lin CS, Tsay JJ. A passive approach for effective detection and localization of region-level video forgery with spatio-temporal coherence analysis. Digital Investigation. 2014;11(2):120–140.

[78] Patwardhan KA, Sapiro G, Bertalmío M. Video inpainting under constrained camera motion. IEEE Transactions on Image Processing. 2007;16(2):545–553.

[79] Criminisi A, Pérez P, Toyama K. Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on image processing. 2004;13(9):1200–1212.

[80] Khodabakhsh A, Ramachandra R, Raja K, Wasnik P, Busch C. Fake Face Detection Methods: Can They Be Generalized? In: 2018 International Conference of the Biometrics Special Interest Group (BIOSIG). IEEE; 2018. p. 1–6.

[81] Wang W, Farid H. Exposing digital forgeries in interlaced and deinterlaced video. IEEE Transactions on Information Forensics and Security. 2007;2(3):438–449.

[82] Kobayashi M, Okabe T, Sato Y. Detecting video forgeries based on noise characteristics. In: Pacific-Rim Symposium on Image and Video Technology. Springer; 2009. p. 306–317.

[83] Chuang WH, Su H, Wu M. Exploring compression effects for improved source camera identification using strongly compressed video. In: Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE; 2011. p. 1953–1956.

[84] Subramanyam AV, Emmanuel S. Video forgery detection using HOG features and compression properties. In: 2012 IEEE 14th International Workshop on Multimedia Signal Processing (MMSP). IEEE; 2012. p. 89–94.

[85] Kaur J, Upadhyay S, Sharma A. A video database for intelligent video authentication. In: Computing, Communication and Automation (ICCCA), 2017 International Conference on. IEEE; 2017. p. 1081–1085.

[86] Aghamaleki JA, Behrad A. Malicious inter-frame video tampering detection in MPEG videos using time and spatial domain analysis of quantization effects. Multimedia Tools and Applications. 2017;76(20):20691–20717.

[87] Qureshi MA, Deriche M. A bibliography of pixel-based blind image forgery detection techniques. Signal Processing: Image Communication. 2015;39:46–74.

[88] Conotter V, O'Brien JF, Farid H. Exposing digital forgeries in ballistic motion. IEEE Transactions on Information Forensics and Security. 2012;7(1):283–296.

[89] Ardizzone E, Mazzola G. A tool to support the creation of datasets of tampered videos. In: International Conference on Image Analysis and Processing. Springer; 2015. p. 665–675.

[90] Qureshi MA, Deriche M, Beghdadi A, Amin A. A critical survey of state-of-the-art image inpainting quality assessment metrics. Journal of Visual Communication and Image Representation. 2017;49:177–191.

[91] Ha T, Lee S, Kim J. Motion compensated frame interpolation by new block-based motion estimation algorithm. IEEE Transactions on Consumer Electronics. 2004;50(2):752–759.

[92] Wexler Y, Shechtman E, Irani M. Space-time completion of video. IEEE Transactions on Pattern Analysis & Machine Intelligence. 2007;p. 463–476.

[93] Shih TK, Tang NC, Tsai JC, Hwang JN. Video motion interpolation for special effect applications. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 2011;41(5):720–732.

[94] Bestagini P, Milani S, Tagliasacchi M, Tubaro S. Local tampering detection in video sequences. In: Multimedia Signal Processing (MMSP), 2013 IEEE 15th International Workshop on. IEEE; 2013. p. 488–493.

[95] Newson A, Almansa A, Fradet M, Gousseau Y, Pérez P. Video inpainting of complex scenes. SIAM Journal on Imaging Sciences. 2014;7(4):1993–2019.

[96] Ebdelli M, Le Meur O, Guillemot C. Video inpainting with short-term windows: application to object removal and error concealment. IEEE Transactions on Image Processing. 2015;24(10):3034–3047.

[97] Lotter W, Kreiman G, Cox D. Unsupervised learning of visual structure using predictive generative networks. arXiv preprint arXiv:151106380. 2015;.

[98] Dar Y, Bruckstein AM. Motion-compensated coding and frame rate up-conversion: Models and analysis. IEEE Transactions on Image Processing. 2015;24(7):2051–2066.

[99] Niklaus S, Mai L, Liu F. Video Frame Interpolation via Adaptive Separable Convolution. In: The IEEE International Conference on Computer Vision (ICCV); 2017. p. 261–270.

[100] Tulyakov S, Liu MY, Yang X, Kautz J. Mocogan: Decomposing motion and content for video generation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 1526–1535.

[101] Walker J, Marino K, Gupta A, Hebert M. The pose knows: Video forecasting by generating pose futures. In: Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE; 2017. p. 3352–3361.

[102] Wang W, Alameda-Pineda X, Xu D, Fua P, Ricci E, Sebe N. Every smile is unique: Landmark-guided diverse smile generation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 7083–7092.

[103] Xiong W, Luo W, Ma L, Liu W, Luo J. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 2364–2373.

[104] Babaeizadeh M, Finn C, Erhan D, Campbell RH, Levine S. Stochastic variational video prediction. arXiv preprint arXiv:171011252. 2017;.

[105] Zhao L, Peng X, Tian Y, Kapadia M, Metaxas D. Learning to forecast and refine residual motion for image-to-video generation. In: European Conference on Computer Vision (ECCV); 2018. p. 387–403.

[106] Yang C, Wang Z, Zhu X, Huang C, Shi J, Lin D. Pose guided human video generation. In: European Conference on Computer Vision (ECCV); 2018. p. 201–216.

[107] Reda FA, Liu G, Shih KJ, Kirby R, Barker J, Tarjan D, et al. Sdc-net: Video prediction using spatially-displaced convolution. In: European Conference on Computer Vision (ECCV); 2018. p. 718–733.

[108] Cai H, Bai C, Tai YW, Tang CK. Deep video generation, prediction and completion of human action sequences. In: European Conference on Computer Vision (ECCV); 2018. p. 366–382.

[109] Cotsaces C, Nikolaidis N, Pitas I. Video shot boundary detection and condensed representation: a review. IEEE signal processing magazine. 2006;23(2):28–37.

[110] Huang CC, Zhang Y, Thing VL. Inter-frame video forgery detection based on multi-level subtraction approach for realistic video forensic applications. In: Signal and Image Processing (ICSIP), 2017 IEEE 2nd International Conference on. IEEE; 2017. p. 20–24.

[111] Zheng L, Sun T, Shi YQ. Inter-frame video forgery detection based on block-wise brightness variance descriptor. In: International Workshop on Digital Watermarking. Springer; 2014. p. 18–30.

[112] Smith E, Basharat A, Anthony Hoogs C, et al. A C3D-Based Convolutional Neural Network for Frame Dropping Detection in a Single Video Shot. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2017. p. 86–94.

[113] Tralic D, Grgic S, Zovko-Cihlar B. Video frame copy-move forgery detection based on Cellular Automata and Local Binary Patterns. In: Telecommunications (BIHTEL), 2014 X International Symposium on. IEEE; 2014. p. 1–4.

[114] Stamm MC, Lin WS, Liu KR. Temporal forensics and anti-forensics for motion compensated video. IEEE Transactions on Information Forensics and Security. 2012;7(4):1315–1329.

[115] Thakur MK, Saxena V, Gupta J. Learning based no reference algorithm for dropped frame identification in uncompressed video. In: Information Systems Design and Intelligent Applications. Springer; 2016. p. 451–459.

[116] He L, Lu W, Jia C, Hao L. Video quality assessment by compact representation of energy in 3D-DCT domain. Neurocomputing. 2017;269:108–116.

[117] Lai WS, Huang JB, Wang O, Shechtman E, Yumer E, Yang MH. Learning blind video temporal consistency. In: European Conference on Computer Vision (ECCV); 2018. p. 170–185.

[118] Joshi V, Jain S. Tampering detection in digital video-a review of temporal fingerprints based techniques. In: Computing for sustainable global development (INDIACom), 2015 2nd International Conference on. IEEE; 2015. p. 1121–1124.

[119] Singh RD, Aggarwal N. Detection of re-compression, transcoding and frame-deletion for digital video authentication. In: Recent Advances in Engineering & Computational Sciences (RAECS), 2015 2nd International Conference on. IEEE; 2015. p. 1–6.

[120] Choi HY, Jang HU, Kim D, Son J, Mun SM, Choi S, et al. Detecting composite image manipulation based on deep neural networks. In: Systems, Signals and Image Processing (IWSSIP), 2017 International Conference on. IEEE; 2017. p. 1–5.

[121] Chen S, Tan S, Li B, Huang J. Automatic detection of object-based forgery in advanced video. IEEE Transactions on Circuits and Systems for Video Technology. 2016;26(11):2138–2151.

[122] Chollet F. Xception: Deep Learning With Depthwise Separable Convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 1251–1258.

[123] Cozzolino D, Poggi G, Verdoliva L. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In: Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security. ACM; 2017. p. 159–164.

[124] Dong C, Deng Y, Change Loy C, Tang X. Compression artifacts reduction by a deep convolutional network. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 576–584.

[125] Cavigelli L, Hager P, Benini L. CAS-CNN: A deep convolutional neural network for image compression artifact suppression. In: Neural Networks (IJCNN), 2017 International Joint Conference on. IEEE; 2017. p. 752–759.

[126] Guo J, Chao H. One-to-many network for visually pleasing compression artifacts reduction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 3038–3047.

[127] Kirmemis O, Bakar G, Murat Tekalp A. Learned Compression Artifact Removal by Deep Residual Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2018. p. 2602–2605.

[128] Yang R, Xu M, Wang Z, Li T. Multi-frame quality enhancement for compressed video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 6664–6673.

[129] Gironi A, Fontani M, Bianchi T, Piva A, Barni M. A video forensic technique for detecting frame deletion and insertion. In: ICASSP; 2014. p. 6226–6230.

[130] Xia M, Yang G, Li L, Li R, Sun X. Detecting video frame rate up-conversion based on frame-level analysis of average texture variation. Multimedia Tools and Applications. 2017;76(6):8399–8421.

[131] Li R, Liu Z, Zhang Y, Li Y, Fu Z. Noise-level estimation based detection of motion-compensated frame interpolation in video sequences. Multimedia Tools and Applications. 2018;77(1):663–688.

[132] Caballero J, Ledig C, Aitken A, Acosta A, Totz J, Wang Z, et al. Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017. p. 2848–2857.

[133] Kupyn O, Budzan V, Mykhailych M, Mishkin D, Matas J. Deblurgan: Blind motion deblurring using conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 8183–8192.

[134] Nah S, Hyun Kim T, Mu Lee K. Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 3883–3891.

[135] Meyer S, Djelouah A, McWilliams B, Sorkine-Hornung A, Gross M, Schroers C. Phasenet for video frame interpolation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 498–507.

[136] Janai J, Güney F, Wulff J, Black MJ, Geiger A. Slow Flow: Exploiting High-Speed Cameras for Accurate and Diverse Optical Flow Reference Data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 2; 2017. p. 6.

[137] Ilan S, Shamir A. A Survey on Data-Driven Video Completion. In: Computer Graphics Forum. vol. 34. Wiley Online Library; 2015. p. 60–85.

[138] Chen J, Kang X, Liu Y, Wang ZJ. Median filtering forensics based on convolutional neural networks. IEEE Signal Processing Letters. 2015;22(11):1849–1853.

[139] Chauhan D, Kasat D, Jain S, Thakare V. Survey on keypoint based copy-move forgery detection methods on image. Procedia Computer Science. 2016;85:206–212.

[140] Li J, Li X, Yang B, Sun X. Segmentation-based image copy-move forgery detection scheme. IEEE Transactions on Information Forensics and Security. 2015;10(3):507–518.

[141] Aksoy Y, Aydin TO, Pollefeys M, Smolić A. Interactive high-quality green-screen keying via color unmixing. ACM Transactions on Graphics (TOG). 2016;35(5):152.

[142] Bideau P, Learned-Miller E. Its moving! A probabilistic model for causal motion segmentation in moving camera videos. In: European Conference on Computer Vision (ECCV). Springer; 2016. p. 433–449.

[143] Xu C, Corso JJ. LIBSVX: A supervoxel library and benchmark for early video processing. International Journal of Computer Vision. 2016;119(3):272–290.

[144] Baig MH, Koltun V, Torresani L. Learning to Inpaint for Image Compression. In: Advances in Neural Information Processing Systems; 2017. p. 1246–1255.

[145] Liu G, Reda FA, Shih KJ, Wang TC, Tao A, Catanzaro B. Image Inpainting for Irregular Holes Using Partial Convolutions. arXiv preprint arXiv:180407723. 2018;.

[146] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Advances in Neural Information Processing Systems; 2014. p. 2672–2680.

[147] Mirza M, Osindero S. Conditional generative adversarial nets. arXiv preprint arXiv:14111784. 2014;.

[148] Isola P, Zhu JY, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017. p. 1125–1134.

[149] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: The IEEE International Conference on Computer Vision (ICCV); 2017. p. 2223–2232.

[150] Luan F, Paris S, Shechtman E, Bala K. Deep Painterly Harmonization. arXiv preprint arXiv:180403189. 2018;.

[151] Hu Y, Wu X, Yu B, He R, Sun Z. Pose-guided photorealistic face rotation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 8398–8406.

[152] Ionescu C, Papava D, Olaru V, Sminchisescu C. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014 jul;36(7):1325–1339.

[153] Xu J, Ni B, Li Z, Cheng S, Yang X. Structure preserving video prediction. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. p. 1460–1469.

[154] He J, Lehrmann A, Marino J, Mori G, Sigal L. Probabilistic video generation using holistic attribute control. In: European Conference on Computer Vision (ECCV); 2018. p. 452–467.

[155] Varol G, Romero J, Martin X, Mahmood N, Black MJ, Laptev I, et al. Learning from synthetic humans. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017. p. 4627–4635.

[156] Richter SR, Hayder Z, Koltun V. Playing for benchmarks. In: The IEEE International Conference on Computer Vision (ICCV); 2017. p. 2213–2222.

[157] Schetinger V, Oliveira MM, da Silva R, Carvalho TJ. Humans are easily fooled by digital images. Computers & Graphics. 2017;68:142–151.

[158] Korshunov P, Marcel S. DeepFakes: a New Threat to Face Recognition? Assessment and Detection. arXiv preprint arXiv:181208685. 2018;.

[159] Rao Y, Ni J. A deep learning approach to detection of splicing and copy-move forgeries in images. In: Information Forensics and Security (WIFS), 2016 IEEE International Workshop on. IEEE; 2016. p. 1–6.

[160] Sutthiwan P, Shi YQ, Zhao H, Ng TT, Su W. Markovian rake transform for digital image tampering detection. In: Transactions on data hiding and multimedia security VI. Springer; 2011. p. 1–17.

[161] Rota P, Sangineto E, Conotter V, Pramerdorfer C. Bad teacher or unruly student: Can deep learning say something in image forensics analysis? In: Pattern Recognition (ICPR), 2016 23rd International Conference on. IEEE; 2016. p. 2503–2508.

[162] Cao H, Kot AC. Identification of recaptured photographs on LCD screens. In: Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. IEEE; 2010. p. 1790–1793.

[163] Bas P, Filler T, Pevnỳ T. Break Our Steganographic System: The Ins and Outs of Organizing BOSS. In: International Workshop on Information Hiding. Springer; 2011. p. 59–70.

[164] Gloe T, Böhme R. The dresden image database for benchmarking digital image forensics. Journal of Digital Forensic Practice. 2010;3(2-4):150–159.

[165] Amerini I, Uricchio T, Ballan L, Caldelli R. Localization of JPEG double compression through multi-domain convolutional neural networks. In: 2017 IEEE Conference on computer vision and pattern recognition workshops (CVPRW). IEEE; 2017. p. 1865–1871.

[166] Schaefer G, Stich M. UCID: An uncompressed color image database. In: Storage and Retrieval Methods and Applications for Multimedia 2004. vol. 5307. International Society for Optics and Photonics; 2003. p. 472–481.

[167] Boroumand M, Fridrich J. Deep learning for detecting processing history of images. Electronic Imaging. 2018;2018(7):1–9.

[168] Huh M, Liu A, Owens A, Efros AA. Fighting fake news: Image splice detection via learned self-consistency. In: European Conference on Computer Vision (ECCV); 2018. p. 101–117.

[169] Shullani D, Al Shaya O, Iuliani M, Fontani M, Piva A. A Dataset for Forensic Analysis of Videos in the Wild. In: International Tyrrhenian Workshop on Digital Communication. Springer; 2017. p. 84–94.

[170] Sanderson C, Lovell BC. Multi-region probabilistic histograms for robust and scalable identity inference. In: International conference on biometrics. Springer; 2009. p. 199–208.

[171] Guan H, Kozak M, Robertson E, Lee Y, Yates AN, Delgado A, et al. MFC Datasets: Large-Scale Benchmark Datasets for Media Forensic Challenge Evaluation. In: 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW). IEEE; 2019. p. 63–72.

[172] Papadopoulou O, Zampoglou M, Papadopoulos S, Kompatsiaris I. A corpus of debunked and verified user-generated videos. Online Information Review. 2019;43(1):72–88.

[173] Kancherla K, Mukkamala S. Novel blind video forgery detection using markov models on motion residue. In: Asian Conference on Intelligent Information and Database Systems. Springer; 2012. p. 308–315.

[174] Hsu CC, Hung TY, Lin CW, Hsu CT. Video forgery detection using correlation of noise residue. In: Multimedia Signal Processing, 2008 IEEE 10th Workshop on. IEEE; 2008. p. 170–174.

[175] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. University of Toronto; 2009.

[176] Kristan M, Matas J, Leonardis A, Vojíř T, Pflugfelder R, Fernandez G, et al. A novel performance evaluation methodology for single-target trackers. IEEE transactions on pattern analysis and machine intelligence. 2016;38(11):2137–2155.

[177] Danelljan M, Hager G, Shahbaz Khan F, Felsberg M. Convolutional Features for Correlation Filter Based Visual Tracking. In: The IEEE International Conference on Computer Vision (ICCV) Workshops; 2015. p. 58–66.

[178] Danelljan M, Robinson A, Khan FS, Felsberg M. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision (ECCV). Springer; 2016. p. 472–488.

[179] Ma C, Huang JB, Yang X, Yang MH. Hierarchical Convolutional Features for Visual Tracking. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 3074–3082.

[180] Wang L, Ouyang W, Wang X, Lu H. Visual Tracking With Fully Convolutional Networks. In: The IEEE International Conference on Computer Vision (ICCV); 2015. p. 3119–3127.

[181] Zhang P, Zhuo T, Huang W, Chen K, Kankanhalli M. Online object tracking based on CNN with spatial-temporal saliency guided sampling. Neurocomputing. 2017;.

[182] Liu H, Zheng Q, Luo M, Zhang D, Chang X, Deng C. How Unlabeled Web Videos Help Complex Event Detection? In: International Joint Conference on Artificial Intelligence (IJCAI); 2017. p. 4040–4046.

[183] Zhao Z, Yang Q, Cai D, He X, Zhuang Y. Video question answering via hierarchical spatio-temporal attention networks. In: International Joint Conference on Artificial Intelligence (IJCAI); 2017. p. 3518–3524.

[184] Yue-Hei Ng J, Hausknecht M, Vijayanarasimhan S, Vinyals O, Monga R, Toderici G. Beyond short snippets: Deep networks for video classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2015. p. 4694–4702.

[185] Kalogeiton V, Ferrari V, Schmid C. Analysing domain shift factors between videos and images for object detection. IEEE transactions on pattern analysis and machine intelligence. 2016;38(11):2327–2334.

[186] Prest A, Leistner C, Civera J, Schmid C, Ferrari V. Learning object class detectors from weakly annotated video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2012. p. 3282–3289.

[187] Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results;.

[188] Coates A, Ng A, Lee H. An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics; 2011. p. 215–223.

[189] Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems; 2014. p. 3320–3328.

[190] LeCun Y, Cortes C, Burges CJ. The MNIST database of handwritten digits; 1998.

[191] Torralba A, Fergus R, Freeman WT. 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE transactions on pattern analysis and machine intelligence. 2008;30(11):1958–1970.

[192] Turkowski K. Filters for common resampling tasks. In: Graphics gems. Academic Press Professional, Inc.; 1990. p. 147–165.

[193] Ravi H, Subramanyam A, Gupta G, Kumar BA. Compression noise based video forgery detection. In: Image Processing (ICIP), 2014 IEEE International Conference on. IEEE; 2014. p. 5352–5356.

[194] Bosse S, Maniry D, Wiegand T, Samek W. A deep neural network for image quality assessment. In: Image Processing (ICIP), 2016 IEEE International Conference on. IEEE; 2016. p. 3773–3777.

[195] Chen YJ, Lin YJ, Hsieh SL. Analysis of Video Quality Variation with Different Bit Rates of H. 264 Compression. Journal of Computer and Communications. 2016;4(05):32.

[196] Sun T, Wang W, Jiang X. Exposing video forgeries by detecting MPEG double compression. In: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on. IEEE; 2012. p. 1389–1392.

[197] Milani S, Bestagini P, Tagliasacchi M, Tubaro S. Multiple compression detection for video sequences. In: Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on. IEEE; 2012. p. 112–117.

[198] He P, Jiang X, Sun T, Wang S. Double compression detection based on local motion vector field analysis in static-background videos. Journal of Visual Communication and Image Representation. 2016;35:55–66.

[199] Elyan E, Gaber MM. A fine-grained Random Forests using class decomposition: an application to medical diagnosis. Neural Computing and Applications. 2016;27(8):2279–2288.

[200] List P, Joch A, Lainema J, Bjontegaard G, Karczewicz M. Adaptive deblocking filter. IEEE transactions on circuits and systems for video technology. 2003;13(7):614–619.

[201] Vazquez-Padin D, Fontani M, Bianchi T, Comesaña P, Piva A, Barni M. Detection of video double encoding with GOP size estimation. In: Information Forensics and Security (WIFS), 2012 IEEE International Workshop on. IEEE; 2012. p. 151–156.

[202] Shanableh T. Detection of frame deletion for digital video forensics. Digital Investigation. 2013;10(4):350–360.

[203] Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.

[204] Zhou P, Han X, Morariu VI, Davis LS. Two-stream neural networks for tampered face detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE; 2017. p. 1831–1839.

[205] Breiman L. Random forests. Machine learning. 2001;45(1):5–32.

[206] Valle R, Cai W, Doshi A. TequilaGAN: How to easily identify GAN samples. arXiv preprint arXiv:180704919. 2018;.

[207] Marra F, Gragnaniello D, Verdoliva L, Poggi G. Do gans leave artificial fingerprints? In: 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE; 2019. p. 506–511.

[208] Fu CM, Alshina E, Alshin A, Huang YW, Chen CY, Tsai CY, et al. Sample adaptive offset in the HEVC standard. IEEE Transactions on Circuits and Systems for Video technology. 2012;22(12):1755–1764.

[209] Sabour S, Frosst N, Hinton GE. Dynamic Routing Between Capsules. arXiv preprint. 2017;.

Appendix A

Integer DCT

This appendix goes into detail on the maths and code behind the Discrete Cosine Transform used in H.264/AVC.

The Discrete Cosine Transform is designed to transform spatial pixel data into the frequency domain where it can be quantised efficiently with minimal visual effect. Here, we discuss the 4x4 DCT as used in H.264. Full details of how the non-integer transform became the H.264/AVC integer-based transform are available in [31].

A two dimensional DCT is used in H.264. This involves a horizontal transform D followed by a vertical transform D^T, followed by element wise multiplication with a scaling matrix E_f as in Equation A.1. The inverse DCT is given in Equation A.2.

$$Y = D_f X D_f^T \otimes E_f \tag{A.1}$$

$$Y = D_i^T (X \otimes E_i) D_i \tag{A.2}$$

$$D_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{A.3}$$

$$D_f = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \tag{A.4}$$

All of the elements of matrix D_f are either ± 1 or ± 2 as seen in Equation A.3. All of the elements of matrix ± 1 or $\pm \frac{1}{2}$ (Equation A.4). The scaling matrix multiplication can be added in to the quantisation function for simplicity. This means that the overall DCT function can be simplified to a process involving only addition, subtraction or bit shifts.

The inverse 4x4 DCT was used to produce Figure 2.4: each of the sixteen 4x4 squares in the diagram represents one high coefficient and all the others set to zero.

Appendix B

The Deep Neural Network

This appendix goes into detail on the maths behind deep neural networks.

Deep neural networks consist of an input layer, and output layer and at least one hidden layer. Each layer consists of a number of neurons. Each neuron is connected to a number of neurons in the previous layer. The output of each neuron is equal to the sum of the weighted inputs plus some bias. This is shown in Equation B.1. In Equation B.1, we are on layer L of the neural network and j is the index into layer L; k is the index into the previous layer $L-1$. jk connects neuron j and neuron k. The output of neuron j in layer L, $a_j^{(L)}$ consists of the sum of the inputs from the previous layer $a_j^{(L-1)}$ multiplied by some calculated weights $w_{jk}^{(L)}$ added to a bias $b_j^{(L)}$ and passed through a non-linear activation function $f()$. Thus:

$$a_j^{(L)} = f\left(b_j^{(L)} + \sum_{k=0}^{K} w_{jk}^{(L)} a_j^{(L-1)}\right) \tag{B.1}$$

The activation function f can be one of a number of non-linear functions, such as: sigmoid ($f(x) = 1/\left(1+e^{-x}\right)$); hyperbolic tangent $f(x) = tanh(x)$; or Rectified Linear Unit or ReLU ($f(x) = max(0, x)$). The output $a_j^{(L)}$ of one layer then becomes the input of the next layer. Deep neural networks can contain multiple hidden layers and the connections between them can be skipped as in skipped networks.

In a supervised neural network, the input data comes in the form of a feature vector, and the output data is a label. The main concept is to adjust the weights and biases such that the output of the last layer of the neural network lies close to the label with minimal error. The process of adjusting the neural network parameters is performed by backpropagation and gradient descent. The label can be expressed as a “one-hot

encoded" vector. This is a vector whose length is equal to the number of possible labels, each index into the vector represents a single labels. Every one-hot encoded vector contains a single 1 at the index representing the correct label and all other elements are set to 0. A neural network outputs a vector of equal size, and the logits of this can be further processed by the softmax function (Equation B.2).

$$softmax(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{B.2}$$

Where z_j is the jth element of the vector z and vector z has K elements. This is repeated for all j in K to get the complete vector. The purpose of the softmax function is to ensure that the sum of all values in z totals 1. This output can then be compared directly with the one-hot encoded label vector and the corresponding difference between them can be used to calculate the cross entropy loss. Other cost functions and loss calculations are also available available, such as L1 loss (mean absolute error) or L2 loss (mean squared error). Cross entropy loss between predicted class $\hat{y}$ and actual label y is given by Equation B.3, where i is the number of classes, which is equal to the length of the label vector.

$$H(\hat{y}, y) = -\sum_{i} \hat{y} \log(y_i) \tag{B.3}$$

Training a network involves minimising loss, and this is done using gradient descent. Stochastic gradient descent and Adam [203], which is an optimisation of stochastic gradient descent, are used in this thesis. For each neuron in a given layer, the partial derivative of the output with respect to each input is calculated, using the chain rule. The partial derivatives of each neuron (index j in layer L, index k in layer $(L-1)$, shown in Equations B.4, B.5 and B.6 are then combined to give the gradient vector.

$$\frac{\partial H}{\partial w_{jk}^{(L)}} = \frac{\partial a_j^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial H}{\partial a_j^{(L)}} \tag{B.4}$$

$$\frac{\partial H}{\partial a_k^{(L-1)}} = \sum_{j=0}^{n_L - 1} \frac{\partial a_j^{(L)}}{\partial a_k^{(L-1)}} \frac{\partial H}{\partial a_j^{(L)}} \tag{B.5}$$

$$\frac{\partial H}{\partial b_{jk}^{(L)}} = \frac{\partial a_j^{(L)}}{\partial b_{jk}^{(L)}} \frac{\partial H}{\partial a_j^{(L)}} \tag{B.6}$$

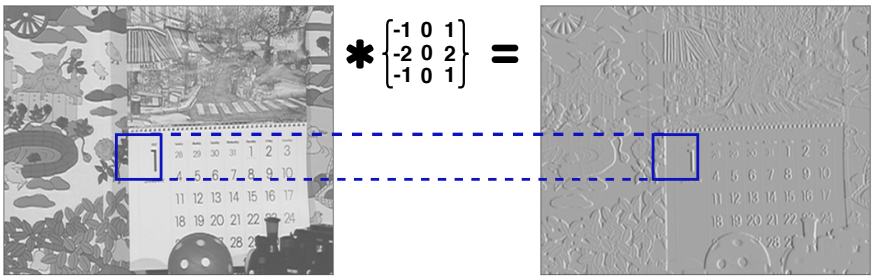


Figure B.1: Convolving an image with a Sobel filter produces a feature map. In this case, the Sobel filter picks out horizontal changes in intensity, thus acting as a detector for vertical edges

This gradient vector gives the direction and magnitude of the change to the weights and bias of the neuron that are required to minimise the loss. This is then back propagated through each layer of the network right to the first layer so that the gradient descent step is calculated for every neuron in the network. Rather than calculate the loss on a single example, mini-batches of examples are given to the network and the loss calculated and averaged. Calculating the loss over mini-batches, rather than the complete training set, allows a series of smaller, less accurate steps to be taken and this is called stochastic gradient descent.

In a fully connected layer, the output of all the neurons of the previous layer $(L-1)$ are connected to each neuron in the current layer, L. A convolutional neural network layer operates slightly differently. In a fully connected layer, the *absolute position* of the data has some influence on the output value. In an image, it is seldom the absolute position of a pixel that dictates its contribution to the output label, but its position *relative* to its neighbours. CNNs account for this spatial invariance by arranging neurons into kernels or filters and convolving these over the image. In this case, convolution is similar to a sliding window and the learned kernels transform the input data into a new representation. Kernelised filtering is also used in Sobel filters, and Figure B.1 illustrates how a feature map is produced from convolving a Sobel kernel with image data. Each layer of a CNN generates a feature map, which is a transformed version of the data from the layer before.

Within a convolutional layer, a kernel is convolved with the input data and results in transformed output data. The resultant output feature map differs from the input size such that:

$$O = (I - F + 2P) * S + 1 \tag{B.7}$$

Where O is the output size, I is the input size, F is the filter or kernel size, P is the

padding. S is the stride or the number of pixels that the kernel moves as it slides over the input data. Padding is optionally applied to the input data to maintain image dimensions through the convolutional process.

As well as fully connected layers and convolutional layers, CNNs also utilise a pooling layer. Again this is a kernelised convolutional layer and is designed to reduce the dimensionality of the feature space. Most commonly, maximum pooling or "maxpool" is used, but average pooling can also be used. To define a maxpooling layer, a kernel size and a stride must be defined, and these control how much the feature maps are downsampled. Although maxpooling layers are used in the networks in this thesis for simplicity, they have been criticised recently in [209]. Maxpooling keeps only the most dominant feature and discards the others, whereas the most recently proposed capsule networks work to route features to appropriate "capsules".

Networks are trained for a given number of epochs, where one epoch is the number of mini-batches necessary to cover the complete dataset. Training time varies according to the task, the data and the hyperparameter settings.

B.1 Whitening

Whitening involves ensuring each image has zero mean and unit variance, which is done as in Equation B.8 where $\bar{x}$ is the mean value of the pixels and s is the maximum of the standard deviation of the pixels and the 1 over the square root of the number of pixels n (as in Equation B.9).

$$\hat{x} = \frac{(x - \bar{x})}{s} \tag{B.8}$$

$$s = max(\sigma, \frac{1}{\sqrt{n}}) \tag{B.9}$$

The datasets used in each of the individual sections of this thesis are detailed in the relevant sections.