ABSTRACT

Title of Dissertation: DEEP LEARNING FOR FORENSICS

Peng Zhou
Doctor of Philosophy, 2020

Dissertation Directed by: Professor Larry Davis
Department of Electrical
and Computer Engineering

The advent of media sharing platforms and the easy availability of advanced photo or video editing software have resulted in a large quantity of manipulated images and videos being shared on the internet. While the intent behind such manipulations varies widely, concerns on the spread of fake news and misinformation is growing. Therefore, detecting manipulation has become an emerging necessity. Different from traditional classification, semantic object detection or segmentation, manipulation detection/classification pays more attention to low-level tampering artifacts than to semantic content. The main challenges in this problem include (a) investigating features to reveal tampering artifacts, (b) developing generic models which are robust to a large scale of post-processing methods, (c) applying algorithms to higher resolution in real scenarios and (d) handling the new emerging manipulation techniques. In this dissertation, we propose approaches to tackling these challenges.

Manipulation detection utilizes both low-level tamper artifacts and semantic contents, suggesting that richer features needed to be harnessed to reveal more evidence. To learn rich features, we propose a two-stream Faster R-CNN network and train it end-to-end to detect the tampered regions given a manipulated image. Experiments on four standard image manipulation datasets demonstrate that our two-stream framework outperforms each individual stream, and also achieves state-of-the-art performance compared to alternative methods with robustness to resizing and compression.

Additionally, to extend manipulation detection from image to video, we introduce VIDNet, Video Inpainting Detection Network, which contains an encoder-decoder architecture with a quad-directional local attention module. To reveal artifacts encoded in compression, VIDNet additionally takes in Error Level Analysis (ELA) frames to augment RGB frames, producing multimodal features at different levels with an encoder.

Besides, to improve the generalization of manipulation detection model, we introduce a manipulated image generation process that creates true positives using currently available datasets. Drawing from traditional work on image blending, we propose a novel generator for creating such examples. In addition, we also propose to further create examples that force the algorithm to focus on boundary artifacts during training. Extensive experimental results validate our proposal.

Furthermore, to apply deep learning models to high resolution scenarios efficiently, we treat the problem as a mask refinement given a coarse low resolution prediction. We propose to convert the regions of interest into strip images and compute a boundary prediction in the strip domain. Extensive experiments on both the public and a newly created high resolution dataset strongly validate our approach.

Finally, to handle new emerging manipulation techniques while preserving performance on learned manipulation, we investigate incremental learning. We propose a multi-model and multi-level knowledge distillation strategy to preserve performance on old categories while training on new categories. Experiments on standard incremental learning benchmarks show that our method improves the overall performance over standard distillation techniques.

DEEP LEARNING FOR FORENSICS

by

Peng Zhou

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Larry Davis, Chair/Advisor
Professor Rama Chellappa
Professor Joseph JaJa
Professor Behtash Babadi
Professor Yang Tao

Dedication

This thesis is dedicated to my parent for their love and support.

Acknowledgments

Graduating during the 2020 pandemic is a special experience, and I would like to thank all the people who helped me survive my PhD. This dissertation could not be completed without the help of them.

First and foremost, I would like to thank my advisor, professor Larry Davis, for his kindness to accept me as his student and his invaluable guidance and support to my research. It is Larry who introduced me to the field of computer vision. It is enjoyable to work with Larry and he always takes care of his students and provides suggestions if needed. Besides, it is honorable to be advised by a famous researcher like him.

Additionally, I want to thank professor Rama Chellappa, professor Joseph JaJa, professor Behtash Babadi and professor Yang Tao for their timely help for serving as my committee members and reviewing all the manuscripts.

I also want to express my gratitude to the Electrical and Computer Engineering department of University of Maryland. My PhD dream would not come true without the admission and course training from it. My gratitude also goes to all the graduate coordinators who helped me submit all types of materials during my PhD period.

My colleagues at the UMIACS are another factor that makes my PhD desirable and thus thank all of them. Thanks Dr. Xintong Han for his help and guidance during my first two years and thanks Dr. Zuxuan Wu for his suggestions and favor during each deadline submission. Also, I am grateful for the assistance from my other co-authors at school including Dr. Vlad Morariu, professor Abhinav Shrivastava, Dr. Sernam Lim, Ning Yu, Dr. Hui Ding, Dr. Mahyar Najibi and Dr. Sirius Chen. It was a pleasure to collaborate with them and their discussion was insightful. Besides that, the time I spent with my colleagues was unforgettable. Special thanks goes to Xintong Han, Zuxuan Wu, Hengduo Li and Shiyi Lan for the fitness we have done together. I also cherish the time spent with Dr. Zhe Wu, Dr. Mingfei Gao, Xitong Yang, Luyu Yang, Jun Wang, Dr. Hao Zhou, Dr. Hongyu Xu and Dr. Pallabi Ghosh. Thank them for all the memorable moments during my PhD period.

My internship experience is also part of my PhD journey and I share the same gratitude to all my mentors for their suggestions and collaborations. Thanks Dr. Long Mai, Dr. Jianming Zhang and Dr. Ning Xu for their suggestions on my incremental learning project; thanks Dr. Brian Price, Dr. Scott Cohen and Dr. Gregg Wilensky for their supportive guidance on the Deepstrip project; thanks Dr. Ran Xu and Dr. Zeyuan Chen for their discussion for the talking face generation project. I have learned a lot and will cherish the time spent with my mentors.

Furthermore, I would like to thank my friends who shared their stories and encouraged me during my PhD. I truly thank my old friend Yiliang Wang, for the holidays we spent together and his suggestions while I felt down. Thanks Ye Jiang and Youru Zhou for their long-lasting friendship and willingness to listen to my unhappiness. Moreover, I am grateful for my 5-year roommate Shengjie Xie and his wife Xi Li who shared foods and TV series with me. Many thanks to other friends in the US including Yi Liu, Jing Huang, Zeyu Zhang, Zhouchen Luo, Xiao Xiao, Shenli Zou, Xiaomin Lin and Zhengyu Lin. I would also express my gratitude to my friends in China, including but not limited to Xiaoqing Wei, Sheng Zhou, Peng Xiao and Wei Xie, who treated me well each time I went back.

Lastly, I owe my deepest thanks to my parents who always trust and stand by me. Thanks for their unconditional love and effort to bring me up. Also thanks for my relatives who took care of me all the time, and I really feel lucky to be a member of my family.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction and Motivation

Recent decades have witnessed a rapid development of deep learning, and it has been applied to various applications including image/video editing, generative model, object recognition and detection. However, the improving result of photo/video editing has raised a lot of concerns about malicious purposes or misinformation. As a result, research comes to light which combines traditional forensics approaches with deep learning to fight against the fake media. In this thesis, we mainly tackle four different challenges to improve forensic detection with deep learning— a) harnessing rich features to find evidence of tampering. b) improving the generalization of deep learning based forensic models. c) exploring efficient solutions to apply deep learning model at different scales. d) extending the application of the learned model to new emerging manipulation techniques.

In Chapter 2, we introduce a two-stream RGB-N network to learn rich features for manipulation detection. One of the two streams is an RGB stream whose purpose is to extract features from the RGB image input to find tampering artifacts like strong contrast difference, unnatural tampered boundaries, and so on. The other is a noise stream that leverages the noise features extracted from a steganalysis rich model filter layer to discover the noise inconsistency between authentic and tampered regions. We then fuse features from the two streams through a bilinear pooling layer to further incorporate spatial co-occurrence of these two modalities.

In Chapter 3, we take the temporal dimension into account and detect the inpainting manipulation within videos. A inpainting detection network VIDNet is then proposed to reveal both spatial and temporal artifacts. The features are learned by our network based on both the compression coefficient artifacts and visual RGB artifacts. After that, the features of these two modalities are further decoded by a Convolutional LSTM to predict masks of inpainted regions. In addition, when detecting whether a pixel is inpainted or not, we present a quad-directional local attention module that borrows information from its surrounding pixels from four directions. Extensive experiments are conducted to validate our approach. We demonstrate that VIDNet outperforms by clear margins alternative inpainting detection methods.

In Chapter 4, we propose combining a generative model to augment training data and thus improve the generalization of the learned model. The network first automatically generates both hard and easy examples, and then segments both boundary artifacts and the interior regions, and finally replaces the predicted artifacts with original regions to refine the predicted result.

In Chapter 5, we study the problem of high resolution boundary refinement to extend the deep learning model to real scenario. We propose transforming the image into an image strip domain to reduce the computation and memory consumption. To detect the target boundary at high resolution, we present a framework with two prediction layers. First, all potential boundaries are predicted as an initial prediction and then a selection layer is used to pick the target boundary and smooth the result. To encourage accurate prediction, a loss which measures the boundary distance in the strip domain is introduced. In addition, we enforce a matching consistency and C0 continuity regularization to the network to reduce false alarms.

In Chapter 6, we further investigate incremental learning to make deep learning models robust to new emerging categories while avoiding forgetting the previous learned knowledge. We leverage all previous model snapshots as the teacher to obtain previous knowledge while trained on new categories. In addition, we incorporate an auxiliary distillation to further preserve knowledge encoded at the intermediate feature levels. To make the model more memory efficient, we adapt mask based pruning to reconstruct all previous models with a small memory footprint.

In Chapter 7, we summarize this dissertation and discuss potential directions for the future research.

Chapter 2: Learning Rich Features for Image Manipulation Detection

2.1 Introduction

With the advances of image editing techniques and user-friendly editing software, low-cost tampered or manipulated image generation processes have become widely available. Among tampering techniques, splicing, copy-move, and removal are the most common manipulations. Image splicing copies regions from an authentic image and pastes them to other images, copy-move copies and pastes regions within the same image, and removal eliminates regions from an authentic image followed by inpainting. Sometimes, post-processing like Gaussian smoothing will be applied after these tampering techniques. Examples of these manipulations are shown in Figure 2.1. Even with careful inspection, humans find it difficult to recognize the tampered regions.

As a result, distinguishing authentic images from tampered images has become increasingly challenging. The emerging research focusing on this topic — image forensics — is of great importance because it seeks to prevent attackers from using their tampered images for unscrupulous business or political purposes. In contrast to

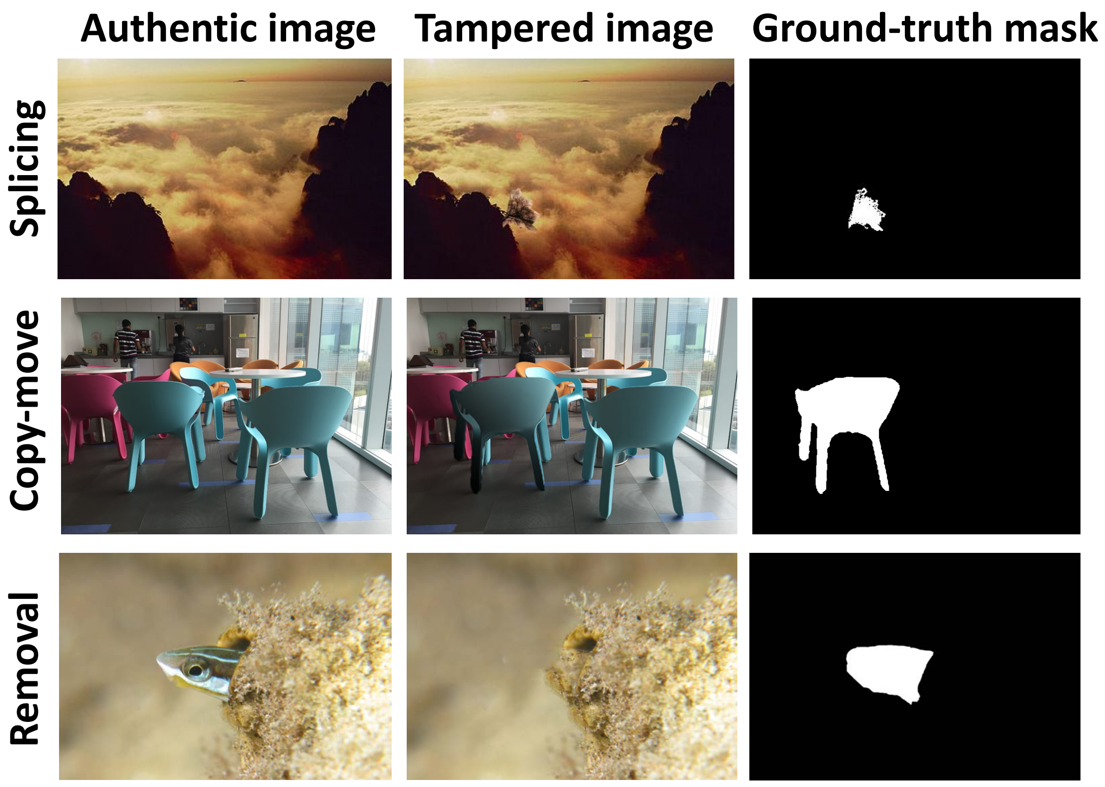


Figure 2.1: Examples of tampered images that have undergone different tampering techniques. From the top to bottom are the examples showing manipulations of splicing, copy-move and removal.

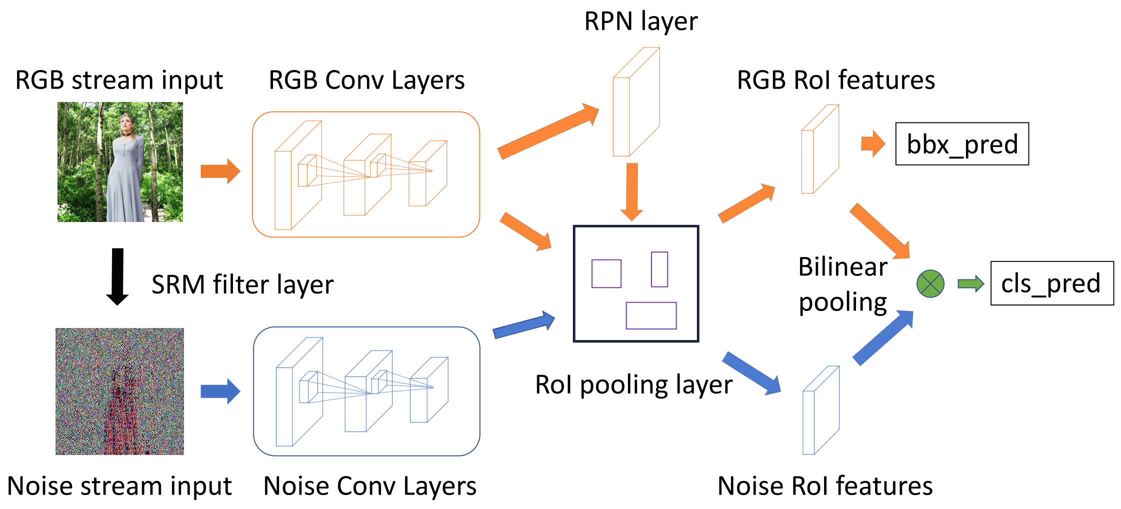


Figure 2.2: Illustration of our two-stream Faster R-CNN network. The RGB stream models visual tampering artifacts, such as unusually high contrast along object edges, and regresses bounding boxes to the ground-truth. The noise stream first obtains the noise feature map by passing input RGB image through an SRM filter layer, and leverages the noise features to provide additional evidence for manipulation classification. The RGB and noise streams share the same region proposals from RPN network which only uses RGB features as input. The RoI pooling layer selects spatial features from both RGB and noise streams. The predicted bounding boxes (denoted as 'bbx_pred') are generated from RGB RoI features. A bilinear pooling [1, 2] layer after RoI pooling enables the network to combine the spatial co-occurrence features from the two streams. Finally, passing the results through a fully connected layer and a softmax layer, the network produces the predicted label (denoted as 'cls_pred') and determines whether predicted regions have been manipulated or not.

current object detection networks [10–15] which aim to detect all objects of different categories in an image, a network for image manipulation detection would aim to detect only the tampered regions (usually objects). We investigate how to adopt object detection networks to perform image manipulation detection by exploring both RGB image content and image noise features.

Recent work on image forensics utilizes clues such as local noise features [16,17] and Camera Filter Array (CFA) patterns [18] to classify a specific patch or pixel [5] in an image as tampered or not, and localize the tampered regions [18–20]. Most of these methods focus on a single tampering technique. A recently proposed architecture [21] based on a Long Short Term Network (LSTM) segments tampered patches, showing robustness to multiple tampering techniques by learning to detect tampered edges. Here, we propose a novel two-stream manipulation detection framework, which not only models visual tampering artifacts (*e.g.*, tampered artifacts near manipulated edges), but also captures inconsistencies in local noise features.

More specifically, we adopt Faster R-CNN [10] within a two-stream network and perform end-to-end training. A summary of our method is shown in Figure 2.2. Deep learning detection models like Faster R-CNN [10] have demonstrated good performance on detecting semantic objects over a range of scales. The Region Proposal Network (RPN) is the component in Faster R-CNN that is responsible for proposing image regions that are likely to contain objects of interest, and it can be adapted for image manipulation detection. For distinguishing tampered regions from authentic regions, we utilize features from the RGB channels to capture clues like visual inconsistencies at tampered boundaries and contrast effect between tampered regions and authentic regions. The second stream analyzes the local noise features in an image.

The intuition behind the second stream is that when an object is removed from one image (the source) and pasted into another (the target), the noise features between the source and target images are unlikely to match. These differences can be partially masked if the user subsequently compresses the tampered image [17,22]. To utilize these features, we transform the RGB image into the noise domain and use the local noise features as the input to the second stream. There are many ways to produce noise features from an image. Based on recent work on steganalysis rich model (SRM) for manipulation classification [16,23], we select SRM filter kernels to produce the noise features and use them as the input channel to the second Faster R-CNN network.

Features from these two streams are then bi-linearly pooled for each Region of Interest (RoI) to detect tampering artifacts based on features from both streams, see Figure 2.2.

Previous image manipulation datasets [4,24–26] contain only several hundred images, not enough to train a deep network. To overcome this, we created a synthetic tampering dataset based on COCO [27] for pre-training our model and then finetuned the model on different datasets for testing. Experimental results of our approach on four standard datasets demonstrate promising performance.

Our contribution is two-fold. First, we show how a Faster R-CNN framework can be adapted for image manipulation detection in a two-stream fashion. We explore two modalities, RGB tampering artifacts and local noise feature inconsistencies, bilinearly pooling them to identify tampered regions. Second, we show that the two streams are complementary for detecting different tampered techniques, leading to improved performance on four image manipulation datasets compared to state-of-the-art methods.

2.2 Related Work

Research on image forensics consists of various approaches to detect the low-level tampering artifacts within a tampered image, including double JPEG compression [22], CFA color array anaylsis [18] and local noise analysis [28]. Specifically, Bianchi *et al.* [22] propose a probabilistic model to estimate the DCT coefficients and quantization factors for different regions. CFA based methods analyze low-level statistics introduced by the camera internal filter patterns under the assumption that the tampered regions disturb these patterns. Goljan *et al.* [18] propose a Gaussian Mixture Model (GMM) to classify CFA present regions (authentic regions) and CFA absent regions (tampered regions).

Recently, local noise features based methods, like the steganalysis rich model (SRM) [23], have shown promising performance in image forensics tasks. These methods extract local noise features from adjacent pixels, capturing the inconsistency between tampered regions and authentic regions. Cozzolino *et al.* [28] explore and demonstrate the performance of SRM features in distinguishing tampered and authentic regions. They also combine SRM features by including the quantization and truncation operations with a Convolutional Neural Network (CNN) to perform manipulation localization [29]. Rao *et al.* [30] use an SRM filter kernel as initialization for a CNN to boost the detection accuracy. Most of these methods focus on specific tampering artifacts and are limited to specific tampering techniques. We also use these SRM filter kernels to extract low-level noise that is used as the input to a Faster R-CNN network, and learn to capture tampering traces from the noise features. Moreover, a parallel RGB stream is trained jointly to model mid- and high-level visual tampering artifacts.

With the success of deep learning techniques in various computer vision and image processing tasks, a number of recent techniques have also employed deep learning to address image manipulation detection. Chen *et al.* [31] add a low pass filter layer before a CNN to detect median filtering tampering techniques. Bayar *et al.* [32] change the low pass filter layer to an adaptive kernel layer to learn the filtering kernel used in tampered regions. Beyond filtering learning, Zhang *et al.* [33] propose a stacked autoencoder to learn context features for image manipulation detection. Cozzolino *et al.* [19] treat this problem as an anomaly detection task and use an autoencoder based on extracted features to distinguish those regions that are difficult to reconstruct as tampered regions. Salloum *et al.* [34] use a Fully Convolutional Network (FCN) framework to directly predict the tampering mask given an image. They also learn a boundary mask to guide the FCN to look at tampered edges, which assists them in achieving better performance in various image manipulation datasets. Bappy *et al.* [21] propose an LSTM based network applied to small image patches to find the tampering artifacts on the boundaries between tampered patches and image patches. They jointly train this network with pixel level segmentation to improve the performance and show results under different tampering techniques. However, only focusing on nearby boundaries provides limited success in different scenarios, *e.g.*, removing the whole object might leave no boundary evidence for detection. Instead, we use global visual tampering artifacts as well as the local noise features to model richer tampering artifacts. We use a two-stream network built on Faster R-CNN to learn rich features for image manipulation detection. The network shows robustness to splicing, copy-move and removal. In addition, the network enables us to make a classification of the suspected tampering techniques.

2.3 Proposed Method

We employ a multi-task framework that simultaneously performs manipulation classification and bounding box regression. RGB images are provided in the RGB stream (the top stream in Figure 2.2), and SRM images in the noise stream (the bottom stream in Figure 2.2). We fuse the two streams through bilinear pooling before a fully connected layer for manipulation classification. The RPN uses the RGB stream to localize tampered regions.

2.3.1 RGB Stream

The RGB stream is a single Faster R-CNN network and is used both for bounding box regression and manipulation classification. We use a ResNet 101 network [35] to learn features from the input RGB image. The output features of the last convolutional layer of ResNet are used for manipulation classification.

The RPN network in the RGB stream utilizes these features to propose RoI for bounding box regression. Formally, the loss for the RPN network is defined as

$$\begin{aligned} L_{RPN}(g_i, f_i) &= \frac{1}{N_{cls}} \sum_i L_{cls}(g_i, g_i^\star) \\ &+ \lambda \frac{1}{N_{reg}} \sum_i g_i^\star L_{reg}(f_i, f_i^\star), \end{aligned} \tag{2.1}$$

where g_i denotes the probability of anchor i being a potential manipulated region in a mini batch, and $g_i^\star$ denotes the ground-truth label for anchor i to be positive. The terms f_i, $f_i^\star$ are the 4 dimensional bounding box coordinates for anchor i and the ground-truth, respectively. L_{cls} denotes cross entropy loss for RPN network and L_{reg} denotes smooth L_1 loss for regression for the proposal bounding boxes. N_{cls} denotes the size of a mini-batch in the RPN network. N_{reg} is the number of anchor locations. The term λ is a hyper-parameter to balance the two losses and is set to 10. Note that in contrast to traditional object detection whose RPN network searches for regions that are likely to be objects, our RPN network searches for regions that are likely to be manipulated. The proposed regions might not necessarily be objects, *e.g.*, the case in the removal tampering process.

2.3.2 Noise Stream

RGB channels are not sufficient to tackle all the different cases of manipulation. In particular, tampered images that were carefully post processed to conceal the splicing boundary and reduce contrast differences are challenging for the RGB stream.

So, we utilize the local noise distributions of the image to provide additional evidence. In contrast to the RGB stream, the noise stream is designed to pay more attention to noise rather than semantic image content. This is novel — while current deep learning models do well in representing hierarchical features from RGB image content, no prior work in deep learning has investigated learning from noise distributions in detection. Inspired by recent progress on SRM features from image forensics [23], we use SRM filters to extract the local noise features (examples shown in Figure 2.3) from RGB images as the input to our noise stream.

In our setting, noise is modeled by the residual between a pixel's value and the estimate of that pixel's value produced by interpolating only the values of neighboring pixels. Starting from 30 basic filters, along with nonlinear operations like maximum and minimum of the nearby outputs after filtering, SRM features gather the basic noise features. SRM quantifies and truncates the output of these filters and extracts the nearby co-occurrence information as the final features. The feature obtained from this process can be regarded as a local noise descriptor [28]. We find that only using 3 kernels can achieve decent performance, and applying all 30 kernels does not give significant performance gain. Therefore, we choose 3 kernels, whose weights are shown in Figure 2.4, and directly feed these into a pre-trained network trained on 3-channel inputs. We define the kernel size of the SRM filter layer in the noise stream to be $5 \times 5 \times 3$. The output channel size of our SRM layer is 3.

The resulting noise feature maps after the SRM layer are shown in the third

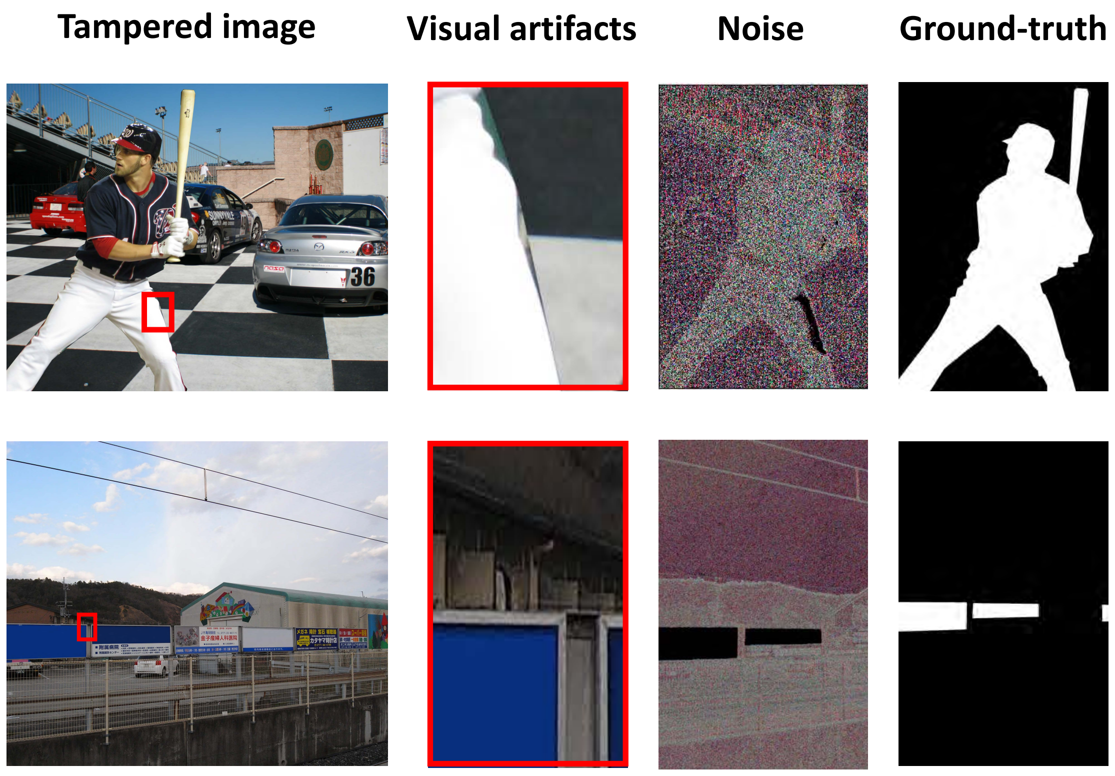


Figure 2.3: Illustration of tampering artifacts. Two examples showing tampering artifacts in the original RGB image and in the local noise features obtained by the SRM filter layer. The second column is the amplified regions for the red bounding boxes in the first column. As shown in the second column, the unnaturally high contrast along the baseball player's edges provides a strong cue about the presence of tampering. The third column shows the local noise inconsistency between tampered regions and authentic regions. In different scenarios, visual information and noise features play a complementary role in revealing tampering artifacts.

$$\frac{1}{4}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 2 & -4 & 2 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \frac{1}{12}\begin{bmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{bmatrix} \quad \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.4: The three SRM filter kernels used to extract noise features.

column of Figure 2.3. It is clear that they emphasize the local noise instead of image content and explicitly reveal tampering artifacts that might not be visible in the RGB channels. We directly use the noise features as the input to the noise stream network. The backbone convolutional network architecture of the noise stream is the same as the RGB stream. The noise stream shares the same RoI pooling layer as the RGB stream. For bounding box regression, we only use the RGB channels because RGB features perform better than noise features for the RPN network based on our experiments (See Table 2.1).

2.3.3 Bilinear Pooling

We finally combine the RGB stream with the noise stream for manipulation detection. Among various fusion methods, we apply bilinear pooling on features from both streams. Bilinear pooling [1], first proposed for fine-grained classification, combines streams in a two-stream CNN network while preserving spatial information to improve the detection confidence. The output of our bilinear pooling layer is $x = f_{RGB}^T f_N$, where f_{RGB} is the RoI feature of the RGB stream and f_N is the RoI feature of the noise stream. Sum pooling squeezes the spatial feature before classification. We then apply signed square root ($x \leftarrow sign(x)\sqrt{|x|}$) and L_2 normalization before forwarding to the fully connected layer.

To save memory and speed up training without decreasing performance, we use compact bilinear pooling as proposed in [2].

After the fully connected and softmax layers, we obtain the predicted class of the RoI regions, as indicated in Figure 2.2. We use cross entropy loss for manipulation classification and smooth L_1 loss for bounding box regression. The total loss function is:

$$L_{total} = L_{RPN} + L_{tamper}(f_{RGB}, f_N) + L_{bbox}(f_{RGB}), \tag{2.2}$$

where L_{total} denotes total loss. L_{RPN} denotes the RPN loss in RPN network. L_{tamper} denotes the final cross entropy classification loss, which is based on the bilinear pooling feature from both the RGB and noise stream. L_{bbox} denotes the final bounding box regression loss. f_{RGB} and f_N are the RoI features from RGB and noise streams. The summation of all terms produces the total loss function.

2.3.4 Implementation Detail

The proposed network is trained end-to-end. The input image as well as the extracted noise features are re-sized so that the shorter length equals to 600 pixels. Four anchor scales with size from 8^2, 16^2, 32^2 to 64^2 are used, and the aspect ratios are 1:2, 1:1 and 2:1. The feature size after RoI pooling is $7 \times 7 \times 1024$ for both RGB and noise streams. The output feature size of compact bilinear pooling is set to 16384. The batch size of RPN proposal is 64 for training and 300 for testing.

Image flipping is used for data augmentation. The Intersection-over Union (IoU) threshold for RPN positive example (potential manipulated regions) is 0.7 and 0.3 for negative example (authentic regions). Learning rate is initially set to 0.001 and then is reduced to 0.0001 after 40K steps. We train our model for 110K steps. At test time, standard Non-Maximum Suppression (NMS) is applied to reduce the redundancy of proposed overlapping regions. The NMS threshold is set to 0.2.

2.4 Experiments

We demonstrate our two stream network on four standard image manipulation datasets and compare the results with state-of-the-art methods. We also compare different data augmentations and measure the robustness of our method to resizing and JPEG compression.

2.4.1 Pre-trained Model

Current standard datasets do not have enough data for deep neural network training. To test our network on these datasets, we pre-train our model on our synthetic dataset. We automatically create a synthetic dataset using the images and annotations from COCO [27]. We use the segmentation annotations to randomly select objects from COCO [27], and then copy and paste them to other images. The training (90%) and testing set (10%) is split to ensure the same background and tampered object do not appear in both training and testing set. Finally, we create 42K tampered and authentic image pairs. We will release this dataset for research use. The output of our model is bounding boxes with confidence scores indicating whether the detected regions have been manipulated or not.

To include some authentic regions in Region of Interest (RoI) for better comparison, We slightly enlarge the default bounding boxes by 20 pixels during training so that both the RGB and noise streams learn the inconsistency between tampered

AP	Synthetic test
RGB Net	0.445
Noise Net	0.461
RGB-N noise RPN	0.472
Noise + RGB RPN	0.620
RGB-N	0.627

Table 2.1: AP comparison on our synthetic COCO dataset. The row is the model architectures, where RGB Net is a single Faster R-CNN using RGB image as input; Noise Net is a single Faster R-CNN using noise feature map as input; RGB-N noise RPN is a two-stream Faster R-CNN using noise features for RPN network. Noise + RGB RPN is a two-stream Faster R-CNN using both noise and RGB features as the input of RPN network. RGB-N is a two-stream Faster R-CNN using RGB features for RPN network.

and authentic regions.

We train our model end-to-end on this synthetic dataset. The ResNet 101 used in Faster R-CNN is pre-trained on ImageNet. We use Average Precision (AP) for evaluation, the metric of which is the same as COCO [27] detection evaluation. We compare the result of the two-stream network with each one of the streams in Table 2.1. This table shows that our two-stream network performs better than each single stream. Also, the comparison among RGB-N, RGB-N using noise features as RPN and RPN uses both features shows that RGB features are more suitable than noise features to generate region proposals.

2.4.2 Testing on Standard Datasets

Datasets. We compare our method with current state-of-the-art methods on NIST Nimble 2016 [25] (NIST16), CASIA [3,26], COVER [4] and Columbia dataset.

- NIST16 is a challenging dataset which contains all three tampering techniques.

Datasets	NIST16	CASIA	Columbia	COVER
Training	404	5123	-	75
Testing	160	921	180	25

Table 2.2: Training and testing split (number of images) for four standard datasets. Columbia is only used for testing the model trained on our synthetic dataset.

The manipulations in this dataset are post-processed to conceal visible traces. They also provide ground-truth tampering mask for evaluation.

• CASIA provides spliced and copy-moved images of various objects. The tampered regions are carefully selected and some post processing like filtering and blurring is also applied. Ground-truth masks are obtained by thresholding the difference between tampered and original images. We use CASIA 2.0 for training and CASIA 1.0 for testing.

• COVER is a relatively small dataset focusing on copy-move. It covers similar objects as the pasted regions to conceal the tampering artifacts (see the second row in Figure 2.1). Ground-truth masks are provided.

• Columbia dataset focuses on splicing based on uncompressed images. Ground-truth masks are provided.

To fine-tune our model on these datasets, we extract the bounding box from the ground-truth mask. We compare with other approaches on the same training and testing split protocol as [21] (for NIST16 and COVER) and [34] (for Columbia and CASIA). See Table 2.2.

Evaluation Metric. We use pixel level F_1 score and Area Under the receiver operating characteristic Curve (AUC) as our evaluation metrics for performance comparison. F_1 score is a pixel level evaluation metric for image manipulation detection, as discussed in [34,36]. We vary different thresholds and use the highest F_1 score as the final score for each image, which follows the same protocol in [34,36]. We assign the confidence score to every pixel in the detected bounding boxes for pixel-level AUC evaluation.

Baseline Models. We compare our proposed method with various baseline models as described below:

• ELA: An error level analysis method [37] which aims to find the compression error difference between tampered regions and authentic regions through different JPEG compression qualities.

• NOI1: A noise inconsistency based method using high pass wavelet coefficients to model local noise [38].

• CFA1:A CFA pattern estimation method [39] which uses nearby pixels to approximate the camera filter array patterns and then produces the tampering probability for each pixel.

• MFCN: A multi-task edge-enhanced FCN based network [34] jointly detecting tampered edges using edge binary masks and tampered regions using tampered region masks.

• J-LSTM: An LSTM based network [21] jointly training patch level tampered edge classification and pixel level tampered region segmentation.

• RGB Net: A single Faster R-CNN network with RGB images as input. *i.e.*, our RGB Faster R-CNN stream.

• Noise Net: A single Faster R-CNN network with noise feature map as input obtained from a SRM filter layer. The RPN network uses noise features in this case.

	NIST16	Columbia	COVER	CASIA
ELA [37]	0.236	0.470	0.222	0.214
NOI1 [38]	0.285	0.574	0.269	0.263
CFA1 [39]	0.174	0.467	0.190	0.207
MFCN [34]	0.571	0.612	-	**0.541**
RGB Net	0.567	0.585	0.391	0.392
Noise Net	0.521	**0.705**	0.355	0.283
Late Fusion	0.625	0.681	0.371	0.397
RGB-N (ours)	**0.722**	0.697	**0.437**	0.408

Table 2.3: F_1 score comparison on four standard datasets. '-' denotes that the result is not available in the literature.

	NIST16	Columbia	COVER	CASIA
ELA [37]	0.429	0.581	0.583	0.613
NOI1 [38]	0.487	0.546	0.587	0.612
CFA1 [39]	0.501	0.720	0.485	0.522
J-LSTM [21]	0.764	-	0.614	-
RGB Net	0.857	0.796	0.789	0.768
Noise Net	0.881	0.851	0.753	0.693
Late Fusion	0.924	0.856	0.793	0.777
RGB-N (ours)	**0.937**	**0.858**	**0.817**	**0.795**

Table 2.4: Pixel level AUC comparison on four standard datasets. '-' denotes that the result is not available in the literature.

- Late Fusion: Direct fusion combining all detected bounding boxes for both RGB Net and noise Net. The confidence scores of the overlapping detected regions from the two streams are set to the maximum one.
- RGB-N: Bilinear pooling of RGB stream and noise stream for manipulation classification and RGB stream for bounding box regression. *i.e.*our full model.

We use the F_1 scores of NOI1, CFA1 and ELA reported in [34] and run the code provided by [36] to obtain the AUC results. The results of MFCN and J-LSTM are replicated from the original literatures as their code is not publicly available.

Table 2.3 shows the F_1 score comparison between our method and the base-

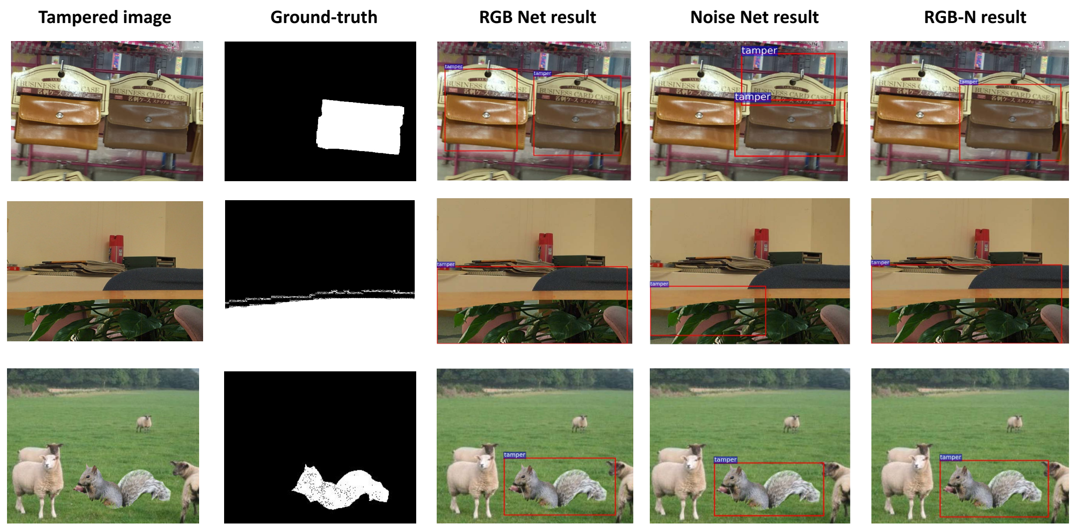


Figure 2.5: Qualitative visualization of results. The top row shows a qualitative result from the COVER dataset. The copy-moved bag confuses the RGB Net, and the noise Net. RGB-N achieves a better detection in this case because it combines the features from the two streams. The middle row shows a qualitative result from the Columbia. The RGB Net produces a more accurate result than noise stream. Taking into account both streams produces a better result for RGB-N. The bottom row shows a qualitative result from the CASIA1.0. The spliced object leaves clear tampering artifacts in both the RGB and noise streams, which yields precise detections for the RGB, noise, and RGB-N networks.

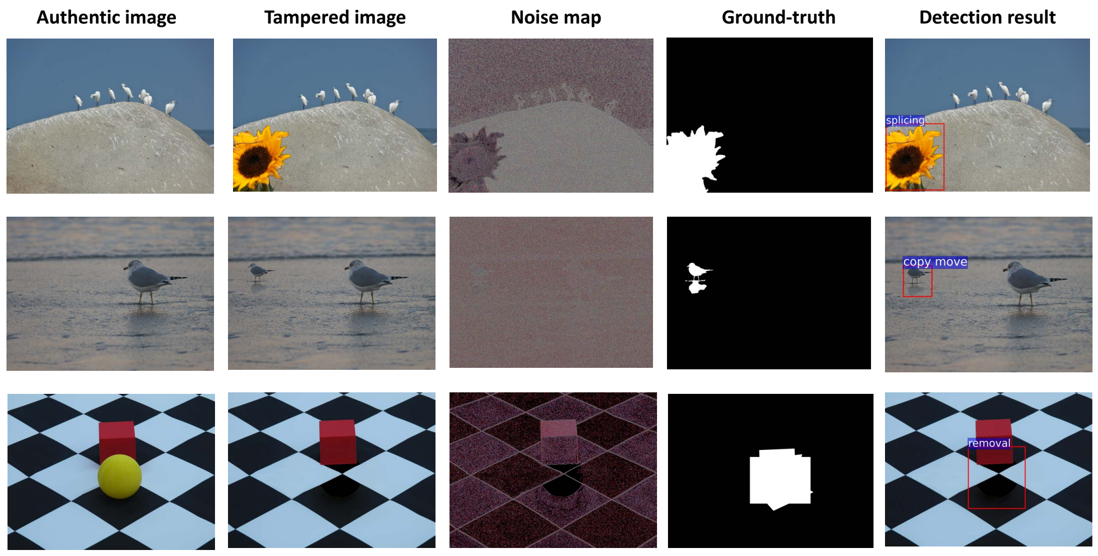


Figure 2.6: Qualitative results for multi-class image manipulation detection on NIST16 dataset. RGB and noise map provide different information for splicing, copy-move and removal. By combining the features from the RGB image with the noise features, RGB-N produces the correct classification for different tamepring techniques.

lines. Table 2.4 provides the AUC comparison. From these two tables, it is clear that our method outperforms conventional methods like ELA, NOI1 and CFA1. This is because they all focus on specific tampering artifacts that only contain partial information for localization, which limits their performance. Our approach outperforms MFCN on Columbia and NIST16 dataset.

One of the reasons our method achieves better performance than J-LSTM is that J-LSTM seeks tampered edges as evidence of tampering, which cannot always detect the entire tampered regions. Also, our method has larger receptive field and captures global context rather than nearby pixels, which helps collect more cues like contrast difference for manipulation classification.

As shown in Table 2.3 and 2.4, our RGB-N network also improves the in-

F_1/AUC	NIST16	COVER	CASIA
Flipping + JPEG	0.712/**0.950**	0.425/0.810	**0.413**/0.785
Flipping + noise	0.717/0.947	0.412/0.801	0.396/0.776
Flipping	**0.722**/0.937	**0.437/0.817**	0.408/**0.795**
No flipping	0.716/0.940	0.312/0.793	0.361/0.766

Table 2.5: Data augmentation comparison. Flipping: image flipping. JPEG: JPEG compression with quality 70. Noise: adding Gaussian noise with variance of 5. Each entry is F_1/AUC score.

dividual streams for all the datasets except Columbia. Columbia only contains uncompressed spliced regions, which preserves noise differences so well that it is sufficient to use only the noise features. This yields satisfactory performance for the noise stream.

For all datasets, late fusion performs worse than RGB-N, which shows the effectiveness of our fusion approach.

Data Augmentation. We compare different data augmentation methods in Table 2.5. Compared with no augmentation, image flipping improves the performance and other augmentation methods like JPEG compression and noise contribute little improvement.

Robustness to JPEG and Resizing Attacks. We test the robustness of our method and compare with 3 methods (whose code is available) in Table 2.6. Our method is more robust to these attacks and outperforms other methods.

2.4.3 Manipulation Technique Detection

The rich feature representation of our network enables it to distinguish between different manipulation techniques as well. We explore manipulation technique de-

JPEG/Resizing	100/1	70/0.7	50/0.5
NOI1	0.285/0.285	0.142/0.147	0.140/0.155
ELA	0.236/0.236	0.119/0.141	0.114/0.114
CFA1	0.174/0.174	0.152/0.134	0.139/0.141
RGB-N	**0.722/0.722**	**0.677/0.689**	**0.677/0.681**

Table 2.6: F_1 score on NIST16 dataset for JPEG compression (with quality 70 and 50) and resizing (with scale 0.7 and 0.5) attacks. Each entry is the F_1 score of JPEG/Resizing.

	Splicing	Removal	Copy-Move	Mean
AP	0.960	0.939	0.903	0.934

Table 2.7: AP comparison on multi-class on NIST16 dataset using the RGB-N network. Mean denotes the mean AP for splicing, removal and copy-move.

tection and analyze the detection performance for all three tampering techniques. NIST16 contains the labels for all three tampering techniques, which enables multi-class image manipulation detection. We change the classes for manipulation classification to be splicing, removal and copy-move so as to learn distinct visual tampering artifacts and noise features for each class. The performance of each tamper class is shown in Table 2.7.

The AP result in Table 2.7 indicates that splicing is the easiest manipulation techniques to detect using our method. This is because splicing has a high probability to produce both RGB artifacts like unnatural edges, contrast differences as well as noise artifacts. Removal detection performance also beats copy-move because the inpainting that follows the removal process has a large effect on the noise features, as shown in Figure 2.3. Copy-move is the most difficult tamper technique for our proposed method. The explanation is that on one hand, the copied regions are from the same image, which yields a similar noise distribution to confuse our noise stream. On the other hand, the two regions generally have the same contrast. Also, the technique would ideally need to compare the two objects to each other (*i.e.*, it would need to find and compare two RoIs at the same time), which the current approach does not do. Thus, our RGB stream has less evidence to distinguish between the two regions.

2.4.4 Qualitative Result

We show some qualitative results in Figure 2.5 for comparison of RGB, noise and RGB-N network in two-class image manipulation detection. The images are selected from the COVER, Columbia and CASIA 1.0. Figure 2.5 provides examples for which our two-stream network yields good performance even if one of the single streams fails (the first and second row in Figure 2.5).

Figure 2.6 shows the results of the RGB-N network on the task of manipulation technique detection task using the NIST16. As is shown in the figure, our network produces accurate results for different tampering techniques.

2.5 Conclusion

We propose a novel network using both an RGB stream and a noise stream to learn rich features for image manipulation detection. We extract noise features by an SRM filter layer adapted from steganalysis literatures, which enables our model to capture noise inconsistency between tampered and authentic regions. We explore the complementary contribution of finding tampered regions from RGB and the noise features of an image. Not surprisingly, the fusion of the two streams leads to improved performance. Experiments on standard datasets show that our method not only detects tampering artifacts but also distinguishes between various tampering techniques. More features, including JPEG compression, will be explored in the future.

Chapter 3: Deep Video Inpainting Detection

3.1 Introduction

Video inpainting, which completes corrupted or missing regions in a video sequence, has achieved impressive progress over the years [40–48]. The ability to produce realistic videos that can be used in applications like video restoration, virtual reality, *etc.*, while appealing, brings significant security concerns at the same time since these techniques can also be used maliciously. By removing objects that could serve as evidence, malicious inpainting can result in serious legal and social implications including swaying a jury, accelerating the spread of misinformation on social platforms, *etc.* Our goal in this work is to develop a framework for detecting inpainted videos constructed with state-of-the-art methods (see Fig. 3.1 for a conceptual overview).

Although there are recent studies on detecting tampered regions in images [6, 49–51], very limited effort has been devoted to video inpainting detection. For image-based manipulation detection, existing approaches either focus on spliced regions or "deepfake"-style face replacement instead of object removal based on inpainting or they are designed specifically for images [52, 53] only and suffer from poor performance on videos. Therefore, it is important to learn robust video representations

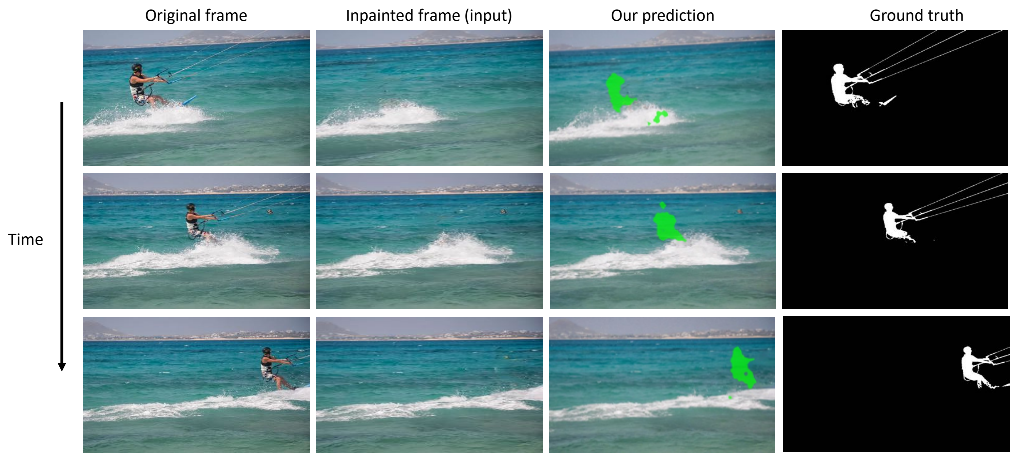


Figure 3.1: **Problem introduction**. Given an inpainted video (second column), we localize the inpainted region both spatially and temporally.

that explore the temporal relationships among frames for video inpainting detection.

In light of this, we introduce VIDNet, a video inpainting detection network, which is an encoder-decoder architecture with a quad-directional local attention module to predict inpainted regions in videos (as is shown in Fig. 3.2). In particular, at each time step, VIDNet takes as inputs the current RGB frame together with its corresponding Error Level Analysis [54] (ELA) frame to the encoder, truncated from a pretrained VGG network [55]. Since video are compressed based on discrete cosine transforms (DCT) and frames extracted are usually stored in JPEG formats, we leverage ELA images as an additional signal to reveal artifacts like compression inconsistency (as is shown in Fig. 3.3). Instead of using ELA images directly, which tends to produce false alarms, we extract features from both ELA and RGB images with the encoder, producing five different multimodal features at different scales, that are further jointly trained for inpainting detection. In addition, given a missing region to fill in, inpainting methods leverage information from surrounding pixels

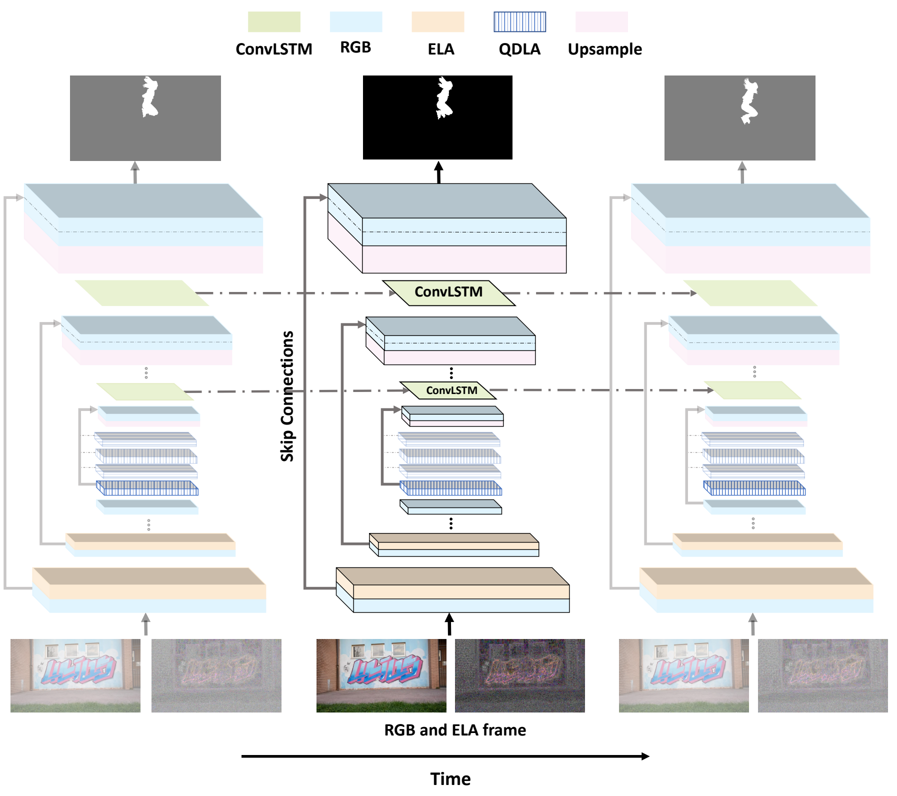


Figure 3.2: **Framework overview**. Given an RGB frame in a video, we first derive its corresponding ELA frame and compute multimodal features at different scales with both frames. We also introduce a quad-directional local attention module (striped) to the last encoded RGB features (colored blue) to explore spatial relationships among pixels from four directions. These encoded features are further input into a multi-layer ConvLSTM (colored green) for decoding, exploiting spatial and temporal relationships explicitly, to produce masks of inpainted regions. See texts for more details.

of the region to make the region coherent spatially. Motivated by this, for RGB features from the last layer of the encoder, we introduce a quad-directional local attention module to attend to the neighbors of a pixel to detect whether that pixel is inpainted or not. This allows us to explicitly model spatial dependencies among different pixels to identify inpainted pixels.

Finally, with multimodal features encoded at different scales, we leverage a four-layer Convolutional LSTM, serving as a decoder for inpainting detection. More specifically, the ConvLSTM at a certain layer not only takes in features from a previous time step but also features upsampled from a coarse level (*i.e.*, a lower decoding layer). In this way both spatial relationships across different scales and temporal dynamics over time are leveraged to produce inpainted masks over time. The framework is trained end-to-end with backpropagation. We conduct experiments on the DAVIS 2016 [56] Dataset and the Free-form Video Inpaiting Dataset [44]. VIDNet successfully detects inpainted regions under all different settings and outperforms by clear margins competing methods. We also show that VIDNet can be generalized to detect out-of-domain inpainted videos that are unseen during training.

Our contributions can be summarized as follows: 1) We target at a relatively new task, to the best of our knowledge, we introduce the first learning based approach for video inpainting detection. 2) We present an end-to-end framework for video inpainting detection, which models spatial and temporal relationships in videos. 3) We leverage multimodal features, *i.e.*, RGB and ELA features, at different scales, for video inpainting detection. 4) We introduce a quad-directional local attention module to explicitly determine if a pixel is inpainted or not by attending to its neighbours.

3.2 Related Work

Video Inpainting. With the advance of recent image inpainting approaches [46–48, 57–62], more recent studies have investigated video inpainting. There are two lines of work — patch based and learning based approaches. For patch based approaches, PatchMatch [63] is a prominent approach which searches for similar patches in the surrounding region iteratively to complete the inpainted region. To achieve better quality, Huang *et al.* [45] explore an optimization based method to match patches and utilize information including color and flow as regularization. On the other hand, learning based approaches have been explored recently. Wang [64] propose a 3D encoder-decoder structure for video inpaining. Afterwards, Xu *et al.* [42] leverage optical flow information to guide the inpainting in videos in both forward and backward passes. Similarly, Kim *et al.* [41] propose to estimate the proceeding flow as additional constraint while completing the missing regions. To maintain more frame pixels, Oh *et al.* [43] use gated convolution to inpaint video frames gradually from the reference frame. Lee *et al.* [40] copy and paste future frames to complete missing details in the current frame. In contrast, our approach detects regions inpainted by these approaches.

Manipulation Detection. There are also approaches focusing on manipulation detection. Most mainly tackle splicing based manipulation and use clues specific to it [19, 51, 65, 66]. In particular, Zhou *et al.* [49] use both RGB and local noise to detect potential regions. Salloum *et al.* [67] rely on boundary artifacts to reveal manipulated regions in a multi-task learning fashion and Zhou *et al.* [68] improve its generalization ability with a generative model. Huh *et al.* [6] use meta-data to find inconsistent patches and Wu *et al.* [50] treat it as anomaly detection to learn features in a self-supervised manner.

More related to our work are methods for image inpainting detection. [53] is a classical approach that searches for similar patches matched by zero-connectivity.

However, high false alarm rates limit their applications in real scenarios. More recently, Zhu *et al.* [69] use CNNs to localize inpainting patches within images. Li *et al.* [52] explore High Pass Filtering (HPF) as the initialization of CNNs for the purpose of distinguishing high frequency noise of natural images from inpainted ones. However, the generalization and robustness is limited as these HPFs are learned given specific inpainting methods. In contrast, we combine both RGB information and ELA features as inputs to VIDNet, and show that our approach generalizes to different inpainting methods. In addition, without temporal guidance, the methods above cannot guarantee temporally consistent prediction like our approach.

3.3 Approach

VIDNet, Video Inpainting Detection Network, is an encoder-decoder architecture (See Fig. 3.2 for an overview the framework) operating on multimodal features to detect inpainted regions. In addition to RGB video frames, VIDNet utilizes Error Level Analysis frames (Sec. 3.3.1) to identify artifacts incurred during the inpainting process. Motivated by the fact that inpaiting methods typically borrow information from neighbouring pixels of the region to be inpainted, we introduce a multi-head local attention module (Sec. 3.3.2) which uses adjacent pixels to discover inpainting traces. Finally, we model the temporal relations among different frames with a ConvLSTM (Sec. 3.3.3). In the following, we describe the components of the model.

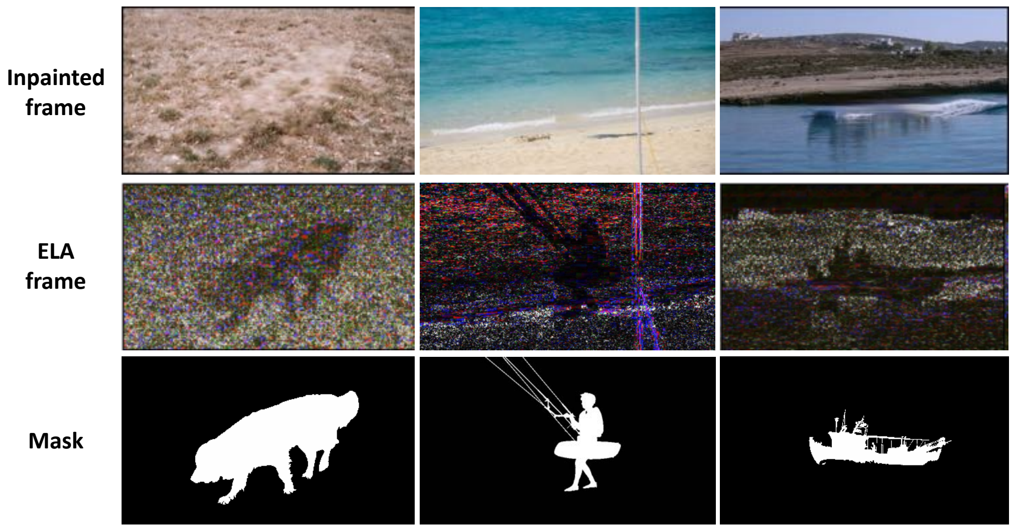


Figure 3.3: **ELA frame example**. From the top to the bottom: the inpainted RGB frame, its corresponding ELA frame, and the ground-truth inpainting mask. The inpainting artifacts, *e.g.*, the dog, person and ship, stand out in ELA space while not easily seen in the RGB space.

3.3.1 Multimodal Features

Learning a mapping directly from an inpainted RGB frame to a mask that encloses the removed object, while feasible, is challenging, since the RGB space is intentionally modified by replacing regions with their surrounding pixels to appear realistic. To mitigate this issue, we additionally augment RGB information with error level analysis features [54] that are designed to reveal regions with inconsistent compression artifacts in compressed JPEG images. Note although videos are usually compressed in MPEG formats, extracted frames are often times stored in the format of JPEG. More formally, an ELA image is defined as:

$$I_{ELA} = ||I - I_{jpg}||_1, \tag{3.1}$$

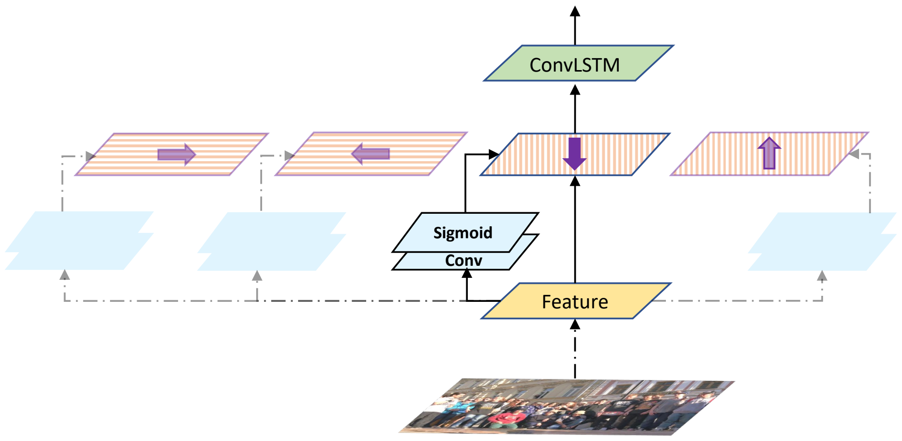


Figure 3.4: **The quad-directional local attention module**. Given RGB features from the last layer of the encoder, we derive attention maps with a quad-directional local attention module. To detect whether a pixel is inpainted or not, the module attends to its neighbors from four directions (left-to-right, up-to-down, right-to-left and down-to-up).

where I_{ELA} is the ELA image, I denotes the original image and I_{jpg} denotes the recompressed JPEG image from the original image.

Fig. 3.3 illustrates the corresponding ELA images of sampled inpainted frames. Although ELA images have been used in forensics applications [36, 66], they tend to create false alarms when other artifacts like *e.g.*, sharp boundaries, are present in the images, which requires ad-hoc judgement to determine whether a region is tampered. So, instead of only using ELA frames, we augment them with RGB frames as inputs to our encoder.

In particular, both the RGB and ELA frame are input to a two-stream encoder. Each stream, based on a VGG encoder, transforms the input image to high-level representations with five layers, yielding 5 feature representations at different scales. At each scale, we normalize the corresponding RGB and ELA features, respectively with ℓ_2 normalization, and then apply one convolutional layer to absorb both features into a unified representation:

$$f_l = ReLU(F(\ [\ ||f_l^{RGB}||_2\ \mid\ ||f_l^{ELA}||_2\ ]))\ \ (l < 5), \tag{3.2}$$

where [||] denotes feature concatenation, f_l denotes the feature at l-th layer. f_l^{RGB}, f_l^{ELA} denote the RGB and ELA features at layer l, respectively. F represents the convolutional layer and $ReLU$ denotes the activation function. The fused representation at each level is further used for decoding. For $l = 5$, we simply use RGB features as we find that high-level ELA features are not helpful.

3.3.2 Quad-Directional Local Attention

Inpainting methods aim to replace a region with pixels from its surrounding areas for photorealistic visual effect. Therefore, when determining whether a pixel is inpainted or not, it is important to examine its surrounding pixels. Inspired by recursive filtering techniques that model pixel relations from four directions for edge-preserving smoothing, we introduce a quad-directional local attention module to explore spatial relations among adjacent pixels.

We learn four attention maps for four directions, left-to-right, right-to-left, top-to-bottom, bottom-to-top, to determine how much information to leverage from the pixels in the corresponding direction based on each map. More specifically, we use $F_{\rightarrow}$, $F_{\leftarrow}$, $F_{\uparrow}$ and $F_{\downarrow}$ to denote functions that derive attention maps for the left-to-right, right-to-left, top-to-bottom and bottom-to-top four directions. In the following, we consider the left-to-right direction for simplicity. Given features f_5

from the last layer of the RGB stream, we first transform the features with $F_{\rightarrow}$ to have the dimension as f_5, and then compute an attention map $A_{\rightarrow}$:

$$A_{\rightarrow} = \sigma(F_{\rightarrow}(f_5; W_{\rightarrow})), \tag{3.3}$$

where $W_{\rightarrow}$ denotes the weights for the convolutional kernel, and σ is the sigmoid function to ensure the attentional weights at each pixel are in the range of $[0, 1]$. Then, for each pixel in the feature map, we obtain information from the surrounding pixels as:

$$f_{5\rightarrow}[k] = (1 - A_{\rightarrow}[k])f_5[k] + A_{\rightarrow}[k]f_5[k-1], \tag{3.4}$$

where k denotes the location of the pixel. Since we are considering attention from the left-to-right direction, $k - 1$ indicates the pixel to the left of k. The current value of pixel k is updated with information from its neighboring pixel, and the weight to balance the contribution $A_{\rightarrow}$ is derived with convolution, which aggregates information from a small grid in the original features. As a result, we attend to a small local region to compute the refined representation. We can derive $f_{5\leftarrow}$, $f_{5\uparrow}$ and $f_{5\downarrow}$ similarly, and thus we have four different refined representations.

Note that the quad-directional attention module is similar in spirit to recursive filtering. However, in standard recursive filtering, a weight matrix, in the form of an edge map [70] or a weighted map [71], is used as our attention map A to guide the filtering to restore images or smooth feature maps. In contrast, our filtering can be considered as a form of self-attention—we derive attention maps by modeling similarities in a local region with convolutions conditioned on input features and the resulting maps are in turn used to refine features, allowing pixels to borrow information by attending to its adjacent pixels. In addition, the motivation of our approach can be seen as the "reverse" process of recursive filtering—in recursive filtering, information from surrounding pixels is diffused to make local regions coherent, whereas we wish to detect inconsistent pixels by attending to a neighboring region.

Furthermore, we compute four refined feature maps for four directions in a parallel way conditioned on the same feature map. An alternative is to generate a single feature representation by sequentially performing attention in four directions, *i.e.*, $f_{5\rightarrow}$ is used as inputs to generate $f_{5\leftarrow}$, and so on and so forth, as in [70]. However, we find through experiments that the parallel multi-head approach offers better results, possibly due to the disentanglement of different directions.

3.3.3 ConvLSTM Decoder

Temporal information like inconsistency in the inpainted region over time is a significant clue for video inpainting detection. To explore temporal relationships among adjacent frames, we use multiple ConvLSTM decoding layers to take features from the encoders and produce predicted detection results, which enables message passing from previous frames. More specifically, the decoder contains four ConvLSTM layers to process features from different spatial scales. At each time step, taking into account both spatial and temporal information, we concatenate together the skipped connected feature of the current frame and the upsampled feature from a lower level, as the inputs to the current ConvLSTM layer. More formally, for the t-th time step, the i-th ($2 <= i <= 4$) ConvLSTM computes the hidden states and cell contents for the $t+1$-th time step as:

$$h_i^{t+1}, c_i^{t+1} = \text{ConvLSTM}_i(\, g_i^t, h_i^t, c_i^t), \tag{3.5}$$

$$g_i^t = [\, U(h_{i-1}^t) \mid f_{6-i}^t \,], \tag{3.6}$$

where h_i^t and c_i^t denote the hidden states and cell states for the i-th ConvLSTM, respectively, and U denotes the function for bilinearly upsampling, which maps the outputs from a lower-level ConvLSTM with smaller feature maps to have the same dimension as the current one. In addition, f_{6-i}^t is the skip connected feature of the frame t from the encoder.

When $i = 1$, the first layer of the ConvLSTM takes features from the last layer of the encoder, *i.e.* f_5 as inputs. Recall that we obtain four refined features based on f_5 with our quad-directional local attention module to identify pixels that are inconsistent with its neighbours from four directions. Thus, we use these refined features as inputs to ConvLSTM$_1$. We input them into the LSTM in the order of $f_{5\rightarrow}$, $f_{5\leftarrow}$, $f_{5\uparrow}$ and $f_{5\downarrow}$ to obtain all the four directional features.

At each time step, we compute g_5^t with Eqn. 3.6 to produce a prediction p^t for each QDLA direction via one convolutional layer. Finally, to explore non-linear relations among these four directional outputs, we fuse them with one additional convolutional layer to form the final prediction. During training, we divide each video into N clips with equal clip length. To encourage more intersection with the binary ground truth mask, we use IoU score [72] as our loss function which is formulated as:

$$L(p, y){=}1 - \frac{\sum_{m=1,w=1}^{H,W} p_m * y_w}{\sum_{m=1,w=1}^{H,W} p_{m,w}{+}\sum_{m=1,w=1}^{H,W} y_{m,w}{-}\sum_{m=1,w=1}^{H,W} p_{m,w} * y_{m,w}{+}\epsilon}, \tag{3.7}$$

where p and y denote the prediction and the binary ground truth mask, respectively. H and W denote the height and width, respectively. ϵ denotes a small number to avoid zero division.

The loss is updated once the ConvLSTM decoder goes through a single video clip to collect temporal information. By exploring spatial and temporal information recurrently, predictions of inpainted regions become more accurate.

3.3.4 Implementation Details

We use PyTorch for implementation. Our model is trained on a NVIDIA GeForce TITAN P6000. The input to the network is resized to 240×427. The length of our video clips is set to 3 frames during training. To extract ELA frames, we recompress the corresponding RGB frames by quality factor 50 and compute their difference. Our feature extraction backbone is VGG-16 [55] for both RGB and ELA features. To increase the generalization ability, we add instance normalization [73] layer to the backbone. The encoder is initialized from VGG-16 model pretrained on ImageNet [74] and the decoder is initialized by Xaiver initialization [75]. We concatenate both RGB and ELA features up to the penultimate encoding layer. Afterwards, the features are passed into one convolutional and normalization layer to reduce the dimension by half to reduce training parameters. The QDLA module is only added to the last encoder layer to extract directional feature information based on ablation results in Sec. 3.4. The decoder is a 4-layer ConvLSTM. We use Adam [76] optimizer with a fixed learning rate of 1×10^{-4} for encoder and 1×10^{-3} for decoder. The optimizer of the encoder and decoder network are updated in an alternating fashion. To avoid overfitting, weight decay with a factor of 5×10^{-5} and 50% dropout [77] are applied. Only random horizontal flipping augmentation is applied during training. We train the whole network end-to-end for 40 epochs with a batch size of 4.

3.4 Experiment

We compare our VIDNet with approaches on manipulation/image inpainting detection in this section to show the advantages of our approach on video inpainting detection. We also analyze the robustness of our approach under different perturbations and show both quantitative and qualitative results.

3.4.1 Experiment setup

Dataset and Evaluation Metrics. Since DAVIS 2016 [56] is the most common benchmark for video inpainting, which consists of 30 videos for training and 20 videos for testing, we evaluate our approach on it for inpainting detection. We generate inpainted videos using SOTA video inpainting approaches — VI [41], OP [43] and CP [40], with the ground truth object mask as reference. To show both the performance and generalization, we choose two out of the three inpainted DAVIS for training and testing, leaving one for additional testing. The training/testing split follows DAVIS default setting. We report the F_1 score and mean Intersection of Union (IoU) to the ground truth mask as evaluation metrics.

We compare our method with the following approaches:

NOI [38]: A traditional approach which aims to find inconsistent noise region as the clue of manipulation. The code for evaluation is from Zampoglou *et al.* [66]. We directly test on the VI, OP and CP test set as it is unsupervised.

CFA [65]: An approach that estimates Camera Filter Array (CFA) and regards the region with different CFA patterns as the manipulated region. We directly test on the VI, OP and CP test set as it is unsupervised.

HPF [52]: A learning based image inpainting detection approach that applies one high pass filter layer as an initialization to reveal high frequency inpainting artifacts. We implement their filter kernel and train the network frame-by-frame from the ImageNet pretrained weights for comparison.

GSR-Net [68]: A generic image manipulation segmentation approach that applies generative models and exploits boundary artifacts to improve the generalization ability. We use their released code and retrain on inpainted DAVIS frame-by-frame for evaluation.

Ours RGB (baseline): Our baseline approach which feeds as input RGB frame only. No QDLA module is applied.

VIDNet-BN (ours): Our batch normalization [78] version approach.

VIDNet-IN (ours): We report this as our main results, which replaces the batch normalization in encoder by instance normalization to improve the generalization across different video inpainting algorithms.

3.4.2 Main Results

Tables 3.1, 3.2 and 3.3 highlight our advantages over other methods. For all the three settings, our IN version outperforms other approaches in both trained and untrained inpainting algorithms, showing the generalization of our approach. Additionally, we show clear improvement over our baseline, indicating the effectiveness of our proposed ELA feature and QDLA module. Comparing across different inpainting algorithms, the performance degrades on the untrained algorithms, indicating a domain shift between trained and untrained inpainting algorithms. However, benefiting from diverse features and more focus on proximity regions, our method still results in better generalization compared with other approaches. Finally, the results indicate that our BN version generally has better performance on the in-domain training inpainting algorithms while IN version shows better generalization on the cross-domain one. Therefore, we provide both results as a trade off between in-domain performance and generalization.

Table 3.1: **mean IoU and F_1 score comparison on inpainted DAVIS.** '*' denotes that the model is trained on these inpainting algorithms.

	VI*		**OP***		**CP**	
Methods	IoU	F1	IoU	F1	IoU	F1
NOI [38]	0.082	0.137	0.090	0.137	0.072	0.132
CFA [65]	0.103	0.142	0.083	0.137	0.076	0.121
HPF [52]	0.456	0.568	0.494	0.615	0.458	0.577
GSR-Net [68]	0.571	0.693	0.500	0.626	0.509	0.634
Ours RGB (baseline)	0.552	0.671	0.456	0.580	0.493	0.625
VIDNet-BN (ours)	**0.620**	**0.726**	**0.749**	**0.833**	**0.670**	**0.775**
VIDNet-IN (ours)	0.585	0.704	0.588	0.707	0.565	0.685

Table 3.2: **mean IoU and F_1 score comparison on inpainted DAVIS.** '*' denotes that the model is trained on these inpainting algorithms.

	VI		**OP***		**CP***	
Methods	IoU	F1	IoU	F1	IoU	F1
NOI [38]	0.082	0.137	0.090	0.137	0.072	0.132
CFA [65]	0.103	0.142	0.083	0.137	0.076	0.121
HPF [52]	0.342	0.444	0.409	0.510	0.676	0.773
GSR-Net [68]	0.302	0.426	0.736	0.818	0.801	0.849
Ours RGB (baseline)	0.308	0.417	0.705	0.773	0.777	0.859
VIDNet-BN (ours)	0.301	0.415	**0.801**	**0.860**	**0.837**	**0.915**
VIDNet-IN (ours)	**0.386**	**0.493**	0.740	0.820	0.810	0.869

3.4.3 Ablation Analysis

We analyze the importance of each key component in our framework and the details are as follows:

Ours ELA: The baseline architecture which only feeds ELA frame as input.

Ours RF edge: Similar to Chen *et al.* [70], we add additional edge branch and apply recursive filter to the final prediction. The output of edge branch is used as the reference to recursive filter layer. The loss function of the edge branch is a weighted binary cross entropy loss.

Table 3.3: **mean IoU and F_1 score comparison on inpainted DAVIS.** '*' denotes that the model is trained on these inpainting algorithms.

	VI*		**OP**		**CP***	
Methods	IoU	F1	IoU	F1	IoU	F1
NOI [38]	0.082	0.137	0.090	0.137	0.072	0.132
CFA [65]	0.103	0.142	0.083	0.137	0.076	0.121
HPF [52]	0.551	0.671	0.186	0.286	0.690	0.796
GSR-Net [68]	0.588	0.703	0.221	0.329	0.700	0.765
Ours RGB (baseline)	0.582	0.689	0.196	0.305	0.753	0.846
VIDNet-BN (ours)	0.578	0.695	0.231	0.323	0.753	0.848
VIDNet-IN (ours)	**0.592**	**0.712**	**0.245**	**0.344**	**0.760**	**0.850**

Table 3.4: **Ablation analysis for each component on our approach.** '*' denotes that the model is trained on these inpainting algorithms.

	VI*		**OP***		**CP**	
Methods	IoU	F1	IoU	F1	IoU	F1
Ours ELA	0.460	0.578	0.509	0.631	0.417	0.546
Ours RGB (baseline)	0.552	0.671	0.456	0.580	0.493	0.625
Ours RF edge	0.540	0.661	0.460	0.591	0.555	0.670
QDLA both features	0.555	0.680	0.580	0.700	0.495	0.635
Ours w/o QDLA	0.559	0.682	0.557	0.681	0.512	0.644
Ours frame-by-frame	0.558	0.683	0.566	0.688	0.532	0.664
Ours w/o ELA	0.568	0.691	0.465	0.595	0.560	0.678
QDLA all layers	0.570	0.693	0.469	0.585	0.564	0.682
VIDNet-IN (ours)	0.585	0.704	0.588	0.707	0.565	0.685

Ours w/o ELA: The baseline applied with QDLA in the last encoder layer. This is our full model without the ELA features.

QDLA both features: Our full model except that the input to QDLA module is the concatenation of both RGB and ELA feature from the 5-th layer.

QDLA all layers: Applying QDLA module to all the 5 encoding feature layers.

Ours frame-by-frame: Instead of training with video clip length of 3, we train our full model frame-by-frame.

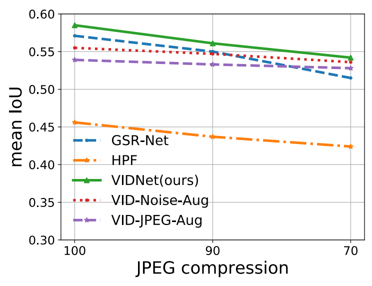
Ours w/o QDLA: Adding ELA feature to the encoder, and concatenating with

RGB feature. The decoder follows baseline which only using temporal information as the additional feature. This is our full model without QDLA module.

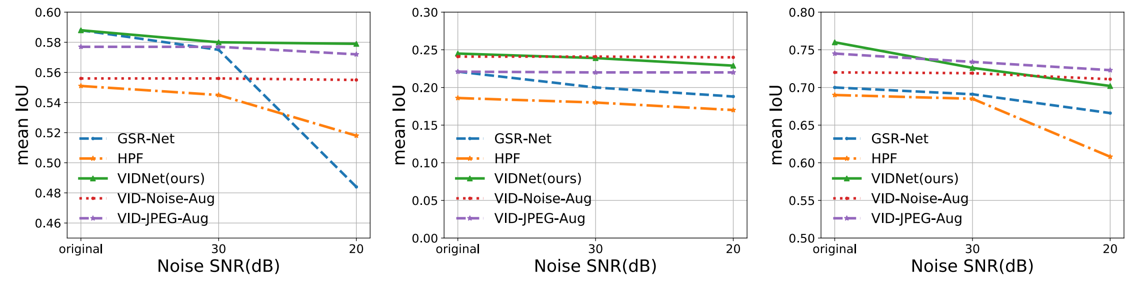
Table 3.4 displays the comparison results. Compared to baseline, the ELA feature alone yields worse performance. This perhaps because the ELA frame also contains other artifacts like sharp boundary, which leads to confusion without proper guidance from RGB contents. Adding QDLA module introduces feature adjacency relationship and thus leads to improvement. However, the higher features are more useful for our QDLA than lower ones when comparing to *QDLA all layers*, and high level ELA features are less helpful than lower ones when comparing with *QDLA both features*. Compared to *Ours RF edge*, our QDLA module (*Ours w/o ELA*) yields better performance because the boundary prediction degrades in video inpainting scenario and thus edge map contains false positives to guide the segmentation branch. In addition, the comparison between *Ours frame-by-frame* and our final model verifies the importance of temporal information in video inpainting detection. Eventually, with QDLA module, ELA feature and temporal information, the performance gets boosted further.

3.4.4 Robustness Analysis

To test the robustness of our approach under noise and JPEG perturbation, we conduct experiments listed in Fig. 3.5. We add Gaussian noise to the input frame with Signal-to-Noise Ratio (SNR) 30 and 20 dB and evaluate on these noisy frames, or recompress test frame with JPEG quality 90 and 70 for perturbation. Moreover,



(a) JPEG perturbation (VI*, OP*, CP)



(b) Noise perturbation (VI*, OP*, CP)

Figure 3.5: Mean IoU comparison under different perturbations. Perturbation in JPEG compression consists of the quality factor with 90 and 70; perturbation in noise consists of SNR 30dB and 20dB. Column from left to right is the result on VI, OP and CP inpainting. '*' denotes that the model is trained on these inpainting algorithms.

to study the effect of specific augmentation in video inpainting detection, we apply noise and JPEG augmentation to our approach and make comparison together. The details of our augmentation is as follow.

VID-Noise-Aug: Randomly apply Gaussian noise with SNR 20 dB to the input frames during training.

VID-JPEG-Aug: Randomly apply JPEG compression with quality factor 90 to the input frames during training.

The robustness of our approach stands out under different perturbations. Compared to other approaches, HPF suffers more from perturbation because more high frequency noises will be introduced. With generative models for augmentation,

Table 3.5: **Mean IoU and F1 score comparison on FVI.** The results are directly tested on FVI dataset, and all the model are trained on VI and OP inpainted DAVIS.

	FVI	
Methods	IoU	F1
NOI [38]	0.062	0.107
CFA [65]	0.073	0.122
HPF [52]	0.205	0.285
GSR-Net [68]	0.195	0.288
Ours RGB (baseline)	0.156	0.223
VIDNet-IN (ours)	**0.257**	**0.367**

GSR-Net shows good robustness. However, our approach outperforms GSR-Net as more modalities of video inpainting clues have been considered. Even though adding noise augmentation results in a small degradation on the initial performance, the robustness to both noise and JPEG perturbation has been improved. Similar observation is made on JPEG augmentation also.

3.4.5 Results on Free-form Video Inpainting Dataset

To further test the performance on different dataset, additional evaluation is provided on Free-form Video Inpainting dataset (FVI). FVI dataset [44] provides 100 test videos, which mostly targets at multi-instance object removal. We directly apply their approach, which leverages 3D gated convolution encoder-decoder architecture for video inpainting, to generate the 100 inpainted videos. To test the generalization of our approach, we directly test the models trained on VI and OP inpainted DAVIS.

Table 3.5 displays the comparison results. Since both the dataset and inpainting approach are different, the performance degrades due to the domain shift. However, compared to other approaches, our method still achieves relatively better

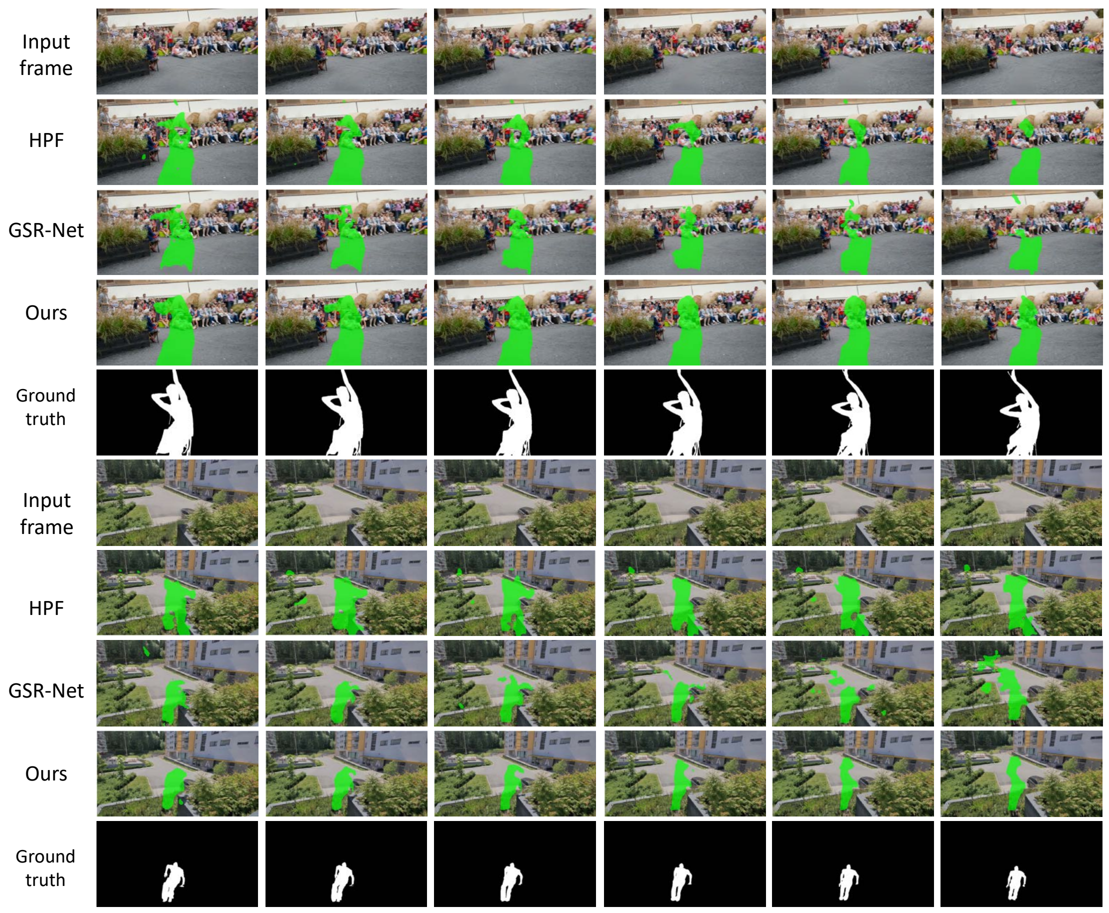


Figure 3.6: Qualitative visualization on DAVIS. The first row shows the inpainted video frame. The second to fourth row indicates the final predictions from different methods. The fifth row is the ground truth.

generalization by a large margin. Also, compared with our baseline model which only uses RGB features, our approach shows clear improvement. This further validates the effectiveness to combine both RGB and ELA features and introduce spatial and temporal information for more evidence.

3.4.6 Qualitative Results

Fig. 3.6 illustrates the visualization of our predictions versus others under the same setting. Thanks to our ELA and RGB features which provide spatial clues, it is clear that our approach is able to obtain a closer prediction to the ground truth than other methods. Specifically, HPF only transfers RGB into noise domain, making it easier to produce false alarm. GSR-Net makes decision frame-by-frame, making the result less temporally consistent. In contrast, with the favor of temporal information, our prediction maintains temporal consistency.

3.5 Conclusions

We introduce learning based video inpainting detection in this paper. To reveal more inpainting artifacts from different domains, we propose to extract both RGB and ELA features and make concatenation. Additionally, we encourage learning from adjacent feature in a self-attended manner by introducing QDLA module. With both the adjacent spatial and temporal information, we make the final prediction through a ConvLSTM based decoder. Our experiments validate the effectiveness of our approach both in-domain and cross-domain. As shown in the results, there still exists a clear gap in the generalization and robustness, making the problem far from being solved. Involving some domain adaption strategies might be a remedy for this issue, which we leave for future research.

Chapter 4: Generate, Segment and Refine: Towards Generic Manipulation Segmentation

4.1 Introduction

Manipulated photos are becoming ubiquitous on social media due to the availability of advanced editing software, including powerful generative adversarial models [79,80]. While such images have been created for a variety of purposes, including memes, satires, etc., there are growing concerns on the abuse of manipulated images to spread fake news and misinformation. To this end, a variety of solutions have been developed towards detecting such manipulated images.

While a number of proposed solutions posed the problem as a classification task [16, 51], where the goal is to classify whether a given image has been tampered with, there is great utility for solutions that are capable of detecting manipulated regions in a given image [6,16,81,82]. In this paper, we similarly treat this problem as a semantic segmentation task and adapt GANs [83] to generate samples to alleviate the lack of training data. The lack of training data has been an ongoing problem for training models to detect manipulated images. Scouring the internet for "real" tampered images [84] is a laborious process that often leads to over-fitting in the

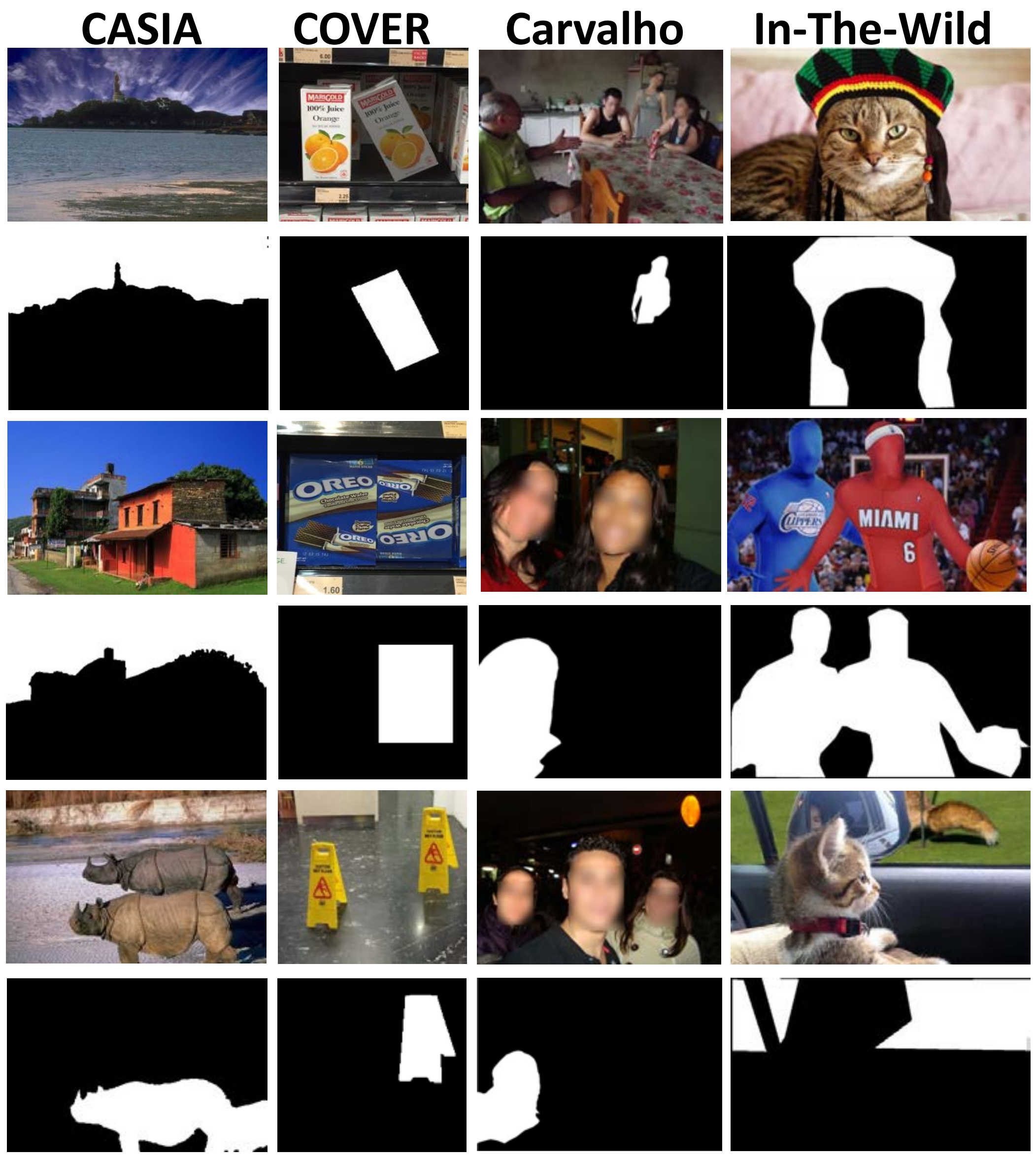


Figure 4.1: **Examples of manipulated images across different datasets.** Columns from left to right are images in CASIA [3], COVER [4], Carvalho [5], and In-The-Wild [6]. The odd rows are manipulated images and the even rows are the ground truth masks. Different datasets contain different distributions (from animals to person), manipulation techniques (from copy-move (the second column) to splicing (the rest columns)) and post-processing methods (from no post-processing to various processes including filtering, illumination, and blurring).

training process. Alternatively, one could employ a self-supervised process, where detected objects in one image are spliced onto another, with the caveat that such

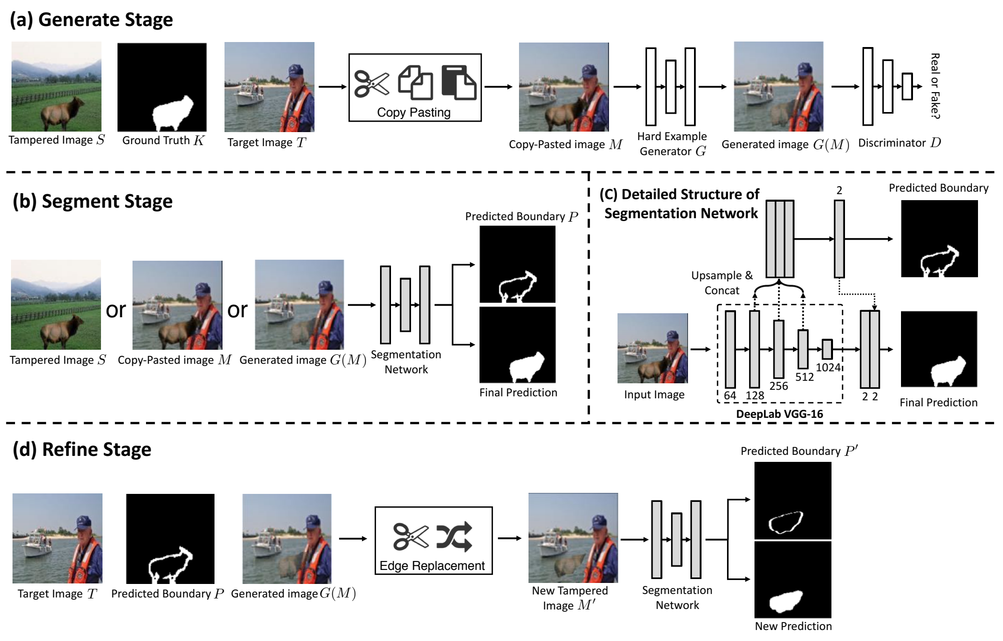


Figure 4.2: **GSR-Net framework overview**. **(a)** Given a tampered image S, an authentic target image T, and the ground truth mask K, the generation stage generates hard example $G(M)$ starting from a simple copy-pasting image M. **(b)** Feeding the training images, copy-pasted images or generated images as input, the segmentation stage learns to segment the boundary artifacts and fill the interior to produce the final prediction. **(c)** The segmentation network concatenates lower level features to predict boundary artifacts and then concatenate back the boundary feature to the segmentation branch for final prediction. **(d)** The refinement stage creates a novel tampered image with new boundary artifacts by replacing the predicted manipulated boundaries of segmentation stage with original authentic regions and learns to make a new prediction.

a process often results in training images that are not realistic. Of course, the best approach for generating training samples is to employ professional labelers to create realistic looking manipulated images, but this remains a very tedious process. It is therefore not surprising that existing datasets [3–6, 26] are often not comprehensive enough to train models that generalize well.

Additionally, in contrast to standard semantic image segmentation, correctly segmenting manipulated regions depends more on visual artifacts that are often created at the boundaries of manipulated regions than on semantic content [21, 49]. Several challenges exist in recognizing these boundary artifacts. First, the space of manipulations is very diverse. One can, for example, do a *copy-move*, which copies and pastes image regions within the same image (the second column in Figure 4.1) , or *splice*, which copies a region from one image and pastes it to another image (the remaining columns in Figure 4.1). Second, a variety of post-processing such as compression, blurring, and various color transformations make it harder to detect boundary artifacts caused by tampering. See Figure 4.1 for some examples. Most existing methods [6, 49, 81, 82] that utilize discriminative features like image metadata, noise models, or color artifacts due to, for example, Color Filter Array (CFA) inconsistencies, have failed to generalize well for these reasons.

This paper introduces a two-pronged approach to (1) address the lack of comprehensive training data, as well as, (2) focus the training process on learning to recognize boundary artifacts better. We adopt GANs for addressing (1), but instead of relying on prior GAN methods [79, 85, 86] that mainly explore image level manipulation, we introduce a novel objective function that optimizes for the realism of the manipulated regions by blending tampered regions in existing datasets to assist segmentation. That is, given an annotated image from an existing dataset, our GAN takes the given annotated regions and optimizes via a blending based objective function to enhance the realism of the regions. Blending has been shown to be effective in creating training images effective for the task of object detection [87], and this forms our main motivation in formulating our GAN.

To address (2), we propose a segmentation and refinement procedure. The segmentation stage localizes manipulated regions by learning to spot boundary artifacts. To further prevent the network from just focusing on semantic content, the refinement stage replaces the predicted manipulation boundaries with authentic background and feed the new manipulated images back to the segmentation network. We will show empirically that the segmentation and refinement has the effect of focusing the model's attention on boundary artifacts during learning (see Table 4.2).

We design an architecture called GSR-Net which includes these three components—a generation stage, a segmentation stage and a refinement stage. The architecture of GSR-Net is shown in Figure 4.2. During training, we alternatively train the generation GAN, followed by the segmentation and refinement stage, which take as input the output of the generation stage as well as images from the training datasets. The additional varieties of manipulation artifacts provided by both the generation and refinement stages produce models that exhibit very good generalization ability. We evaluate GSR-Net on four public benchmarks and show that it performs better to state-of-the-art methods. Experiments with two different post-processing attacks further demonstrate the robustness of GSR-Net. In summary, the contributions of this paper are 1) A framework that augments existing datasets in a way that specifically addresses the main weaknesses of current approaches without requiring new annotations efforts; 2) Introducing a generation stage with a novel objective function based on blending for generating images effective for training models to detect tampered regions; 3) Introducing a novel refinement stage that encourages the learning of boundary artifacts inherent in manipulated regions, which, to the best of our knowledge, no prior work in this field has utilized to help training.

4.2 Related Work

Image Manipulation Segmentation. [81] train a network to find JPEG compression discrepancies between manipulated and authentic regions. [16,49] harness noise features to find inconsistencies within a manipulated image. [6] treat the problem as anomaly segmentation and use metadata to locate abnormal patches. The features used in these works are based on the assumption that manipulated regions are from a different image, which is not the case in copy-move manipulation. However, our method directly focuses on general artifacts in the RGB channel without specific feature extraction and thus can be applied to copy-move segmentation. More related works from [82] and [21] show the potential of boundary artifacts in different image manipulation techniques. These methods are sources of motivation for us to exploit boundary artifacts as a strong cue for detecting manipulations. [21] design a Long Short-Term Memory (LSTM) [88] based network to identify RGB boundary artifacts at both the patch and pixel level. [82] adopt a Multi-task Fully Convolutional Network (MFCN) to manipulation segmentation by providing both segmentation and edge annotations. Instead of applying hole filling on edge prediction to do late fusion, our segmentation stage early fuses edge information with segmentation branch to improve segmentation results.

GAN Based Image Editing. GAN based image editing approaches have witnessed a rapid emergence and impressive results have been demonstrated recently [85, 86, 89–91]. Prior and concurrent works force the output of GAN to be conditioned on input images through extra regression losses (for example, ℓ_2 loss) or discrete labels. However, these methods manipulate the whole images and do not fully explore region based manipulation. In contrast, our GAN manipulates minor regions and fits better for manipulation segmentation where minor regions have been manipulated. A more related work [89] generates natural composite images using both scene parsing and harmonized ground truth. Even though it targets at region manipulation, experimental results show that our method performs better in terms of assisting segmentation.

Adversarial Training. Discriminative feature learning has motivated recent research on adversarial training on several tasks. [92] propose a simulated and unsupervised learning approach which utilizes synthetic images to generate realistic images. An online hard negative generation network [93] boosts the performance on occluded and deformed objects. [94] investigate an adversarial erasing approach to learn dense and complete semantic segmentation. [95] propose an adversarial shadow attenuation network to make correct predictions on hard shadow examples. However, their approaches are difficult to adapt to manipulation segmentation because they either generate whole synthetic images or leave artifacts on erased regions. In contrast, we replace manipulated regions with original ones so that the replaced regions become authentic.

4.3 Approach

We describe the GSR-net in details in the following sections. A key to the generation is the utilization of a GAN with a loss function central around using blending to optimize for producing realistic training images. The segmentation and refinement stage are specially designed to single out boundaries of the manipulated regions in order to guide the training process to pay extra attention to boundary artifacts.

4.3.1 Generation

Generator. Referring to Figure 4.2 (a), the generator is given as input both copy-pasted images and ground truth masks. To prepare the input images, we start with the training samples in manipulation datasets (for example, CASIA 2.0 [3]). Given a training image S, the corresponding ground truth binary mask K and an authentic target image T from a clean dataset (for example, COCO [27]), we first create a simple copy-pasted image M by taking S as foreground and T as background:

$$M = K \odot S + (1 - K) \odot T, \tag{4.1}$$

where $\odot$ represents pointwise multiplication.

In Poisson blending [96], the final value of pixel i in the manipulated regions is

$$
\begin{aligned}
b_i = \arg\min_{b_i} & \sum_{s_i \in S, \mathcal{N}_i \subset S} ||\nabla b_i - \nabla s_i||_2 \\
& + \sum_{s_i \in S, \mathcal{N}_i \not\subset S} ||b_i - t_i||_2, \qquad (4.2)
\end{aligned}
$$

where ∇ denotes the gradient, $\mathcal{N}_i$ is the neighborhood (for example, up, down, left and right) of the pixel at position i, b_i is the pixel in the blended image B, s_i is the pixel in S and t_i is the pixel in T.

Similar to Poisson blending, we optimize the generator to blend neighborhoods in the resulting image that now contains copy-pasted regions and background regions. A key part of our loss function enforces the shapes of the tampered regions, while maintaining the background regions. To maintain background regions, we utilize ℓ_1 loss to reconstruct the background:

$$
L_{\text{bg}} = \frac{1}{N_{\text{bg}}} \sum_{t_i \in T, k_i = 0} ||m_i - t_i||_1, \qquad (4.3)
$$

where N_{bg} is the total number of pixels in the background, m_i is the pixel in M and k_i is the value in mask K at position i. To maintain the shape of manipulated regions, we apply a Laplacian operator to the pasted regions and reconstruct the gradient of this region to match the source region:

$$
L_{\text{grad}} = \frac{1}{N_{\text{fg}}} \sum_{s_i \in S, k_i = 1} ||\Delta m_i - \Delta s_i||_1, \qquad (4.4)
$$

where Δ denotes the Laplacian operator and N_{fg} is the total number of pixels in pasted regions. To further constrain the shape of pasted regions, we add an additional edge loss as denoted by

$$L_{\text{edge}} = \frac{1}{N_{\text{edge}}} \sum_{s_i \in S, e_i = 1} ||m_i - s_i||_1, \tag{4.5}$$

where N_{edge} is the number of boundary pixels and e_i is the value of the edge mask at position i, which is obtained by the absolute difference between a dilation and an erosion on K. To generate realistic manipulated images, we add an adversarial loss L_{adv}, as explained below, that serves to encourage the generator to produce increasingly realistic images as the training progresses.

Discriminator. In our discriminator, a crucial detail to point out is that the manipulated regions are typically occupying only a small area in the image. Hence, it is beneficial to restrict the GAN discriminator's attention to the structure in local images patches. This is reminiscent of "PatchGAN" [79] that only penalizes structure at the scale of patches. Similar to PatchGAN, our discriminator applies a final fully convolutional layer at a patch scale of $N \times N$. The discriminator distinguishes the authentic image T as real and the generated image $G(K, M)$ as fake by maximizing:

$$\begin{aligned} L_{\text{adv}} = &\, \mathbb{E}_T[\log(D(K, T))] \\ &+ \mathbb{E}_M[1 - \log(D(K, G(K, M)))], \end{aligned} \tag{4.6}$$

where K is concatenated with $G(K, M)$ or T as the input to the discriminator (we do not show K in the discriminator input in Figure 4.2 (a) for simplicity).

The final loss function of the generator is given as

$$L_G = L_{\text{bg}} + \lambda_{\text{grad}} L_{\text{grad}} + \lambda_{\text{edge}} L_{\text{edge}} + \lambda_{\text{adv}} L_{\text{adv}}, \tag{4.7}$$

where λ_{grad}, λ_{edge}, and λ_{adv} are parameters which control the importance of the corresponding loss terms. Conditioned on this constraint, the generator preserves background and texture information of pasted regions while blending the manipulated regions with the background, which can be applied to generate both splicing and copy-move examples. Also, it can be potentially utilized to generate removal examples by setting λ_{grad} and λ_{edge} to zero, and thus the generator learns to inpaint the missing regions, creating images with removal manipulation.

4.3.2 Segmentation

For segmentation, we simply adopt the publicly available VGG-16 [97] based DeepLab model [70] to include boundary information. The network structure is depicted in Figure 4.2 (c), consisting of a boundary branch predicting the manipulated boundaries and a segmentation branch predicting the interior. In particular, to enhance attention on boundary artifacts, we introduce boundary information by subtracting the erosion from the dilation of the binary ground truth mask to obtain the boundary mask. We then predict this boundary mask through concatenating bilinearly up-sampled intermediate features and passing them to a 1×1 convolutional

Dataset	**Carvalho**		**In-The-Wild**		**COVER**		**CASIA**	
Metrics	MCC	F1	MCC	F1	MCC	F1	MCC	F1
NOI [38]	0.255	0.343	0.159	0.278	0.172	0.269	0.180	0.263
CFA [39]	0.164	0.292	0.144	0.270	0.050	0.190	0.108	0.207
MFCN [82]	0.408	0.480	-	-	-	-	0.520	0.541
RGB-N [49]	0.261	0.383	0.290	0.424	0.334	0.379	0.364	0.408
EXIF-consistency [6]*	0.420	0.520	0.415	0.504	0.102	0.276	0.127	0.204
DeepLab (baseline)	0.343	0.420	0.352	0.472	0.304	0.376	0.435	0.474
GSR-Net (ours)	**0.462**	**0.525**	**0.446**	**0.555**	**0.439**	**0.489**	**0.553**	**0.574**

Table 4.1: **MCC and F_1 score comparison on four standard datasets.** '-' denotes that the result is not available in the literature. * Our method is 1600 times faster than EXIF-consistency.

layer to form the boundary branch. Finally, we concatenate the output features of the boundary branch with the up-sampled features of the segmentation branch. Empirically, we noticed such multi-task learning helps the generalization of the final model. Only the segmentation branch output after boundary feature concatenation is used for evaluation during inference. During training, we select the copy-pasted examples M, generated examples $G(M)$ and training samples S in the dataset as input to the segmentation network which provides a larger variety of manipulation. The loss function of the segmentation network is an average, two class softmax cross entropy loss.

4.3.3 Refinement

The goal of the refinement stage is to draw attention to the boundary artifacts during training, taking into account the fact that boundary artifacts play a more pivotal role than semantic content in detecting manipulations [21,49]. While we may be able to employ prior erasing based adversarial mining methods [93,94], they are not suitable for our purpose because it will introduce artifacts on the erased regions that should become authentic background. Instead, the refinement stage utilizes the prediction of the segmentation stage to produce new boundary artifacts through replacing with original regions. As illustrated in Figure 4.2 (d), given an authentic target image T in which the manipulated regions was inserted, the manipulated image M (which could also be the generated image $G(M)$), and the manipulated boundary prediction P by the segmentation stage, we replace the pixels in predicted boundaries by the authentic regions in T and create a novel manipulated image:

$$M' = T \odot P + M \odot (1 - P), \tag{4.8}$$

where M' is the novel manipulated image with new boundary artifacts. The corresponding segmentation ground truth now becomes

$$K' = K - K \odot P, \tag{4.9}$$

where K' is the new manipulated mask for M'. The new boundary artifact mask can be extracted in the same way as the previous step. Notice that the refinement stage utilizes the target images T to help training, providing more side information to spot the artifacts. Taking as input the new manipulated images, the same segmentation network in Figure 4.2 (c) then learns to predict the new manipulated boundaries and interior regions.

In addition to augment boundary artifacts, the refinement stage also mines the hard examples during training. Since the refinement stage is based on predictions from the previous stage, hard examples where the manipulation regions are not predicted remain the same after the replacing operation. As a result, these hard examples weight more during training after feeding back to the segmentation network.

Similar to [94], multiple refinement operations are possible and there is a tradeoff between training time and performance. However, the difference is that the segmentation network in the refinement stage shares weights with that in the segmentation stage. The weight sharing enables us to use a single segmentation network at inference. As a result, the network learns to focus more attention on boundary artifacts with no additional cost at inference time.

4.4 Experiments

We evaluate the performance of GSR-Net on four public benchmarks and compare it with the state-of-the-art methods. We also analyze its robustness under several attacks.

4.4.1 Datasets and Experiment Setting

Datasets. We evaluate our performance on four datasets — In-The-Wild [6], COVER [4], CASIA 1.0 [26] and Carvalho [5].

Evaluation Metrics. We use pixel-level F1 score and MCC as the evaluation metrics when comparing to other approaches. For fair comparison, following the same measurement as [6, 49, 82], we vary the prediction threshold to get binary prediction mask and report the optimal score over the whole dataset.

4.4.2 Main Results

In this section, We present our results for the task of manipulation segmentation. We fine-tune our model on CASIA 2.0 from the ImageNet pre-trained model and test directly the performance on the aforementioned four datasets. We compare with methods described below:

- **NoI** [38]: A noise inconsistency method which predicts regions as manipulated where the local noise is inconsistent with authentic regions. We use the released code [36] for evaluation.
- **CFA** [39]: A CFA based method which estimates the internal CFA pattern of the camera for every patch in the image and segments out the regions with anomalous CFA features as manipulated regions. The evaluation code is public available [36].
- **RGB-N** [49]: A two-stream Faster R-CNN based approach which combines features from the RGB and noise channel to make the final prediction. We train the model on CASIA 2.0 using the code provided by the authors.
- **MFCN** [82]: A multi-task FCN based method which harnesses both an edge mask and segmentation mask for manipulation segmentation. Hole filling is applied for the edge branch to make the prediction. The final decision is the intersection of the two branches. We directly report the results from the paper since the code is not publicly available.
- **EXIF-consistency** [6]: A self-consistency approach which utilizes metadata to

learn features useful for manipulation localization. The prediction is made patch by patch and post-processing like mean-shift [98] is used to obtain the pixel-level manipulation prediction. We use the code provided by the authors for evaluation.

- **DeepLab**: Our baseline model which adopts DeepLab VGG-16 model to manipulation segmentation task. No generation, boundary branch or refinement stage is added.

- **GSR-Net**: Our full model combining generation, segmentation and refinement for manipulation segmentation.

The final results, presented in Table 4.1, highlight the advantage of GSR-Net. For supervised methods [49, 82], we train the model on CASIA 2.0 and evaluate on all the four datasets. For other unsupervised methods [6,38,39], we directly test the model on all datasets. GSR-Net outperforms other approaches by a large margin on COVER, suggesting the advantage of our network on copy-move manipulation. Also, GSR-Net has an improvement on In-The-Wild, CASIA 1.0 and Carvalho. Additionally, in terms of computation time, EXIF-consistency takes 1600 times more computation (80 seconds for an 800×1200 image on average) than ours (0.05s per image). Compared to boundary artifact based methods, our GSR-Net outperforms MFCN by a large margin, indicating the effectiveness of the generation and refinement stages. In addition to that, no hole filling is required since our approach does not perform late fusion with the boundary branch, but utilizing boundary artifacts to guide the segmentation branch instead.

Our method outperforms the baseline model by a large margin, showing the effectiveness of the proposed generation, segmentation and refinement stages.

Dataset	Carvalho	In-the-Wild	COVER	CASIA
DeepLab	0.420	0.472	0.376	0.474
DL + CP	0.446	0.504	0.410	0.503
DL + G	0.460	0.524	0.434	0.506
DL + DIH	0.384	0.421	0.342	0.420
DL + CP + G	0.472	0.528	0.444	0.507
GS-Net	0.515	0.540	0.455	0.545
GSR-Net	0.525	0.555	0.489	0.574

Table 4.2: **Ablation analysis on four datasets.** Each entry is the F1 score tested on individual dataset.

4.4.3 Ablation Analysis

We quantitatively analyze the influence of each component in GSR-Net in terms of F1 score.

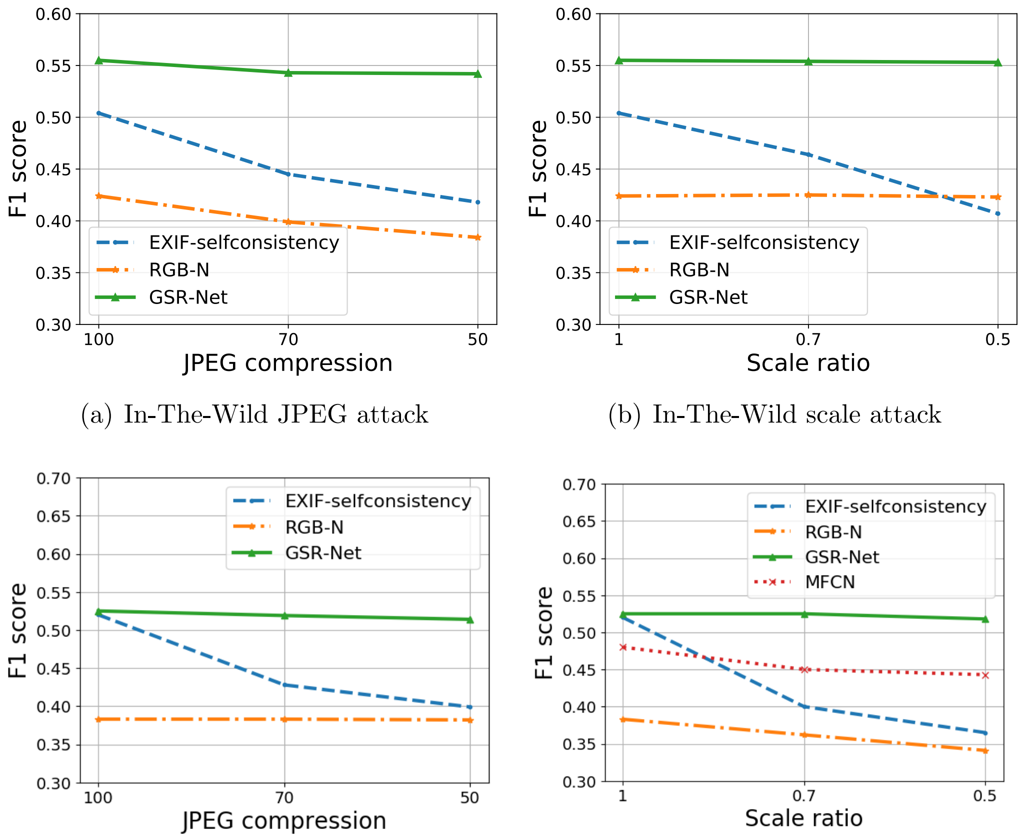
• **DL + CP**: DeepLab VGG-16 model with just the segmentation output, using simple copy-pasted (no generator) and CASIA 2.0 images during training.

• **DL + G**: DeepLab VGG-16 model with just the segmentation output, using generated and CASIA 2.0 images during training.

• **DL + DIH**: DeepLab VGG-16 model with just the segmentation output, using the images generated from [89] and CASIA 2.0 images during training. We adapt deep image harmonization (DIH) network for the generation stage as it also manipulate regions.

• **DL + CP + G**: DeepLab VGG-16 model with just the segmentation output, using both copy-pasted, generated and CASIA 2.0 images during training.

• **GS-Net**: Generation and segmentation network with boundary artifact guided



(a) In-The-Wild JPEG attack (b) In-The-Wild scale attack

(c) Carvalho JPEG attack (d) Carvalho scale attack

Figure 4.3: **Analysis of robustness under different attacks.** Attacks with JPEG compression consists of quality factors of 70 and 50; scale attacks use scaling ratios of 0.7 and 0.5. **(a)** JPEG compression attacks on In-The-Wild. **(b)** Scale attacks on In-The-Wild. **(c)** JPEG compression attacks on Carvalho. **(d)** Scale attacks on Carvalho.

manipulation segmentation. No refinement stage is incorporated.

The results are shown in Table 4.2. Starting from our baseline model, simply adding copy-pasted images (**DL + CP**) achieves improvement due to broadening the manipulation distribution. In addition, replacing copy-pasted images with generated images (**DL + G**) also shows improvement compared to **DL + CP** on all the datasets as it refines the boundary from naive copy-pasting. As expected, adding both copy-pasted images and generated hard examples (**DL + CP + G**) is more

Dataset	Carvalho	In-The-Wild	COVER	CASIA
CP + S	0.343	0.430	0.351	0.242
CP + G + S	0.354	0.441	0.355	0.270
CP + GSR	**0.418**	**0.479**	**0.381**	**0.331**

Table 4.3: **F1 score manipulation segmentation comparison trained with COCO annotations.**

useful because the network has access to a larger distribution of manipulation.

Compared to applying deep harmonization network (**DL + DIH**), our generation approach (**DL + G**) performs better as it aligns well with the natural process of manipulation and has a larger variety of manipulation.

The results also indicate the impact of boundary guided segmentation network. Directly predicting segmentation (**DL + CP + G**) does not explicitly learn manipulation artifacts, and thus has limit generalization ability compared to **GS-Net**, which uses the boundary features as side information. Furthermore, **GSR-Net** boosts the performance on **GS-Net** since the refinement stage introduces new boundary artifacts.

4.4.4 Robustness to Attacks

We apply both JPEG compression and image scaling attacks to test images of In-The-Wild and Carvalho datasets. We compare GSR-Net with RGB-N [49], EXIF-selfconsistency [6] using their publicly available code, and MFCN [82] using the numbers reported in their paper. Figure 4.3 shows the results, which indicates our approach yields more stable performance than prior methods.

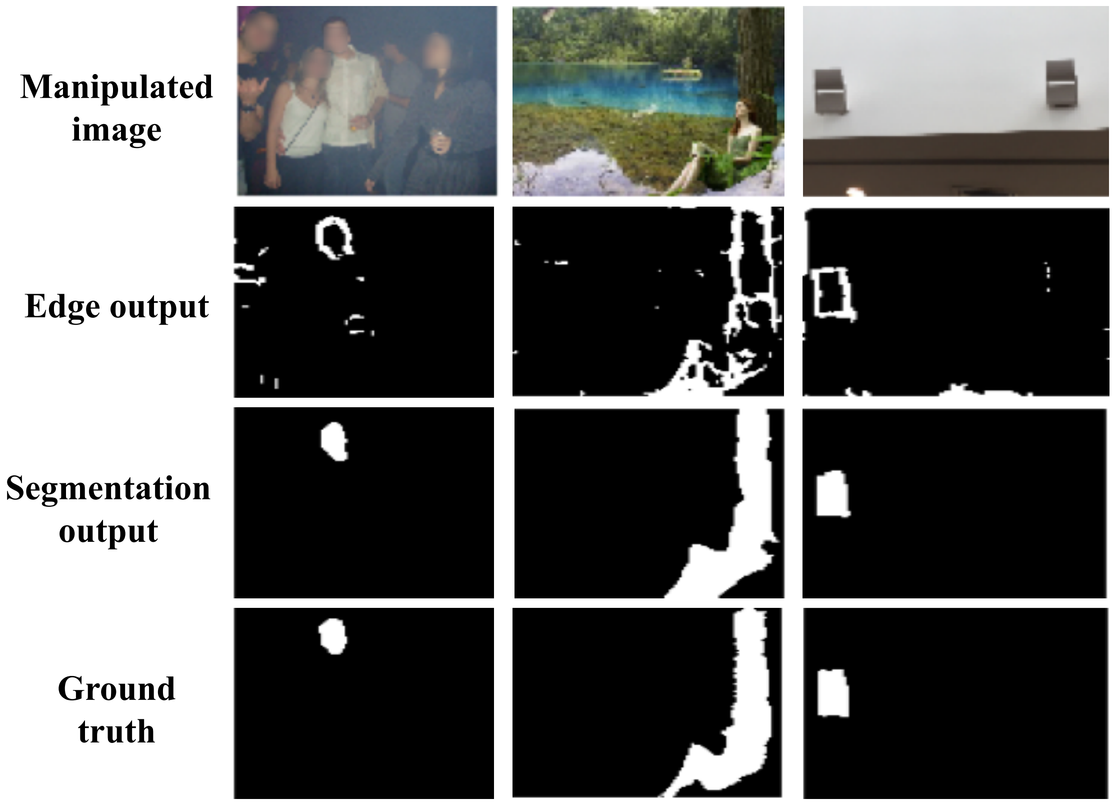


Figure 4.4: **Qualitative visualization.** The first row shows manipulated images on different datasets. The second indicates the final manipulation segmentation prediction. The third row illustrates the output of boundary artifacts branch. The last row is the ground truth.

4.4.5 Segmentation with COCO Annotations

This experiment shows how much gain our model achieves without using the manipulated images in CASIA 2.0. Instead of carefully manipulated training data, we only utilize the object annotations in COCO to create manipulated images. We compare the result of using different training data as follows:

- **CP + S**: Only using copy-pasted images to train the segmentation network.
- **CP + G + S**: Using both copy-pasted and generated images.
- **CP + GSR**: Using copy-pasted images and generated images. The refinement

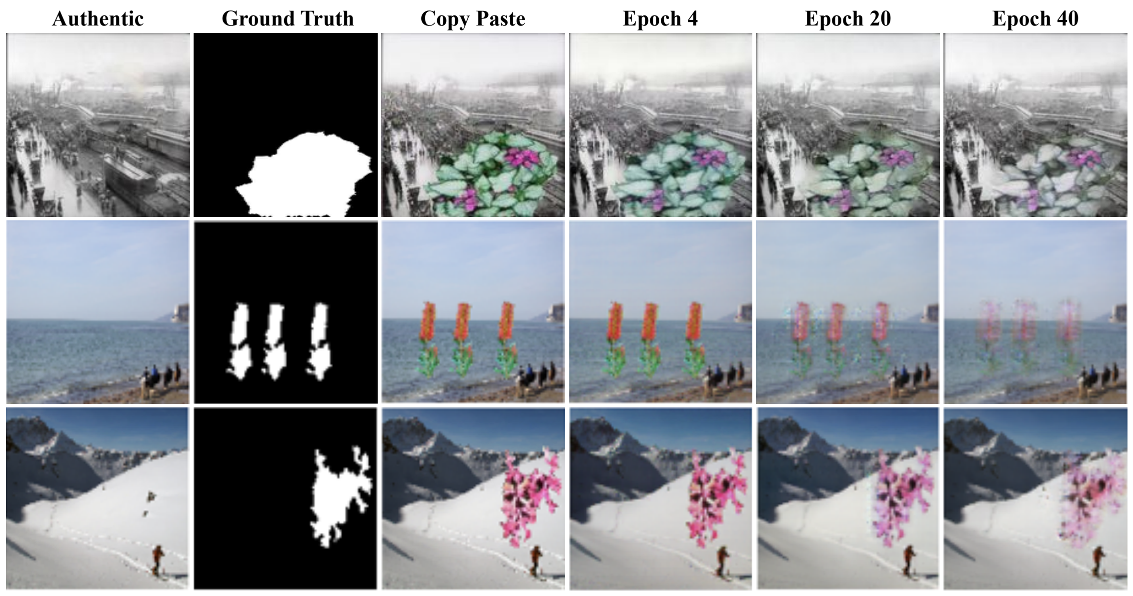


Figure 4.5: **Qualitative visualization of the generation network.** The first two columns show the authentic background and manipulation mask. As the number of epochs increases, the manipulated region matches better with the background and thus boundary artifacts are harder to identify.

stage is applied.

Results are presented in Table 4.3. The performance using only copy-pasted images (**CP + S**) on the four datasets indicates that our network truly learns boundary artifacts. Also, the improvement after adding generated images (**CP + G + S**) shows that our generation network provides useful manipulation examples that increases generalization. Last, the refinement stage (**CP + GSR**) boosts performance further by encouraging the network to spot new boundary artifacts.

4.4.6 Qualitative Results

Generation Visualization. We illustrate some visualizations of the generation network in Figure 4.5. It is clear that the generation network learns to match the pasted region with background during training. As a result, the boundary artifacts are becoming subtle and the generation network produces harder examples for the segmentation network.

Segmentation Results. We present qualitative segmentation results on four datasets in Figure 4.4. Unsurprisingly, the boundary branch outputs the potential boundary artifacts in manipulated images and the other branch fills in the interior based on the predicted manipulated boundaries. The examples indicate that our approach deals well with both splicing and copy-move manipulation based on the manipulation clues from the boundaries.

4.5 Conclusion

We propose a novel segmentation framework that firstly utilizes a generation network to enable generalization across variety of manipulations. Starting from copy-pasted examples, the generation network generates harder examples during training. We also design a boundary artifact guided segmentation and refinement network to focus on manipulation artifacts rather than semantic content. Furthermore, the segmentation and refinement stage share the same weights, allowing for much faster inference. Extensive experiments demonstrate the generalization ability and effectiveness of GSR-Net on four standard datasets and show state-of-the-art performance. The manipulation segmentation problem is still far from solved due to the large variation of manipulations and post-processing methods. Including more manipulation techniques in the generation network could potentially boost the generalization ability of the existing model and is part of our future research.

Chapter 5: DeepStrip: High Resolution Boundary Refinement

5.1 Introduction

Boundary detection is a well-studied problem and fundamental for human recognition [99, 100]. Recent decades have witnessed considerable effort to improve the boundary quality of an object that has been detected [101–109] or segmented [110–114]. Consequently, it is not difficult to separate object of interests from backgrounds with precise boundaries utilizing these methods. While current learning based boundary detection algorithms are usually computed on low resolution (LR) images (0.04-0.25 million pixels), most photos taken these days are much larger, ranging from cell phone size (8-16 million pixels) to professional camera size (16-400 million pixels). Most methods are not designed for images of this size and the excessive computation they require, and most machine learning based methods cannot process them due to memory constraints. Given a precise low resolution prediction, a workaround would be to directly apply upsampling to reach high resolution (HR). Nevertheless, this usually yields poor quality results because the semantic contents in the HR image are not considered. (See Figure 5.1.)

Most research in boundary detection focuses on improving the boundary quality in LR through introducing more semantic information [8, 115, 116] or human

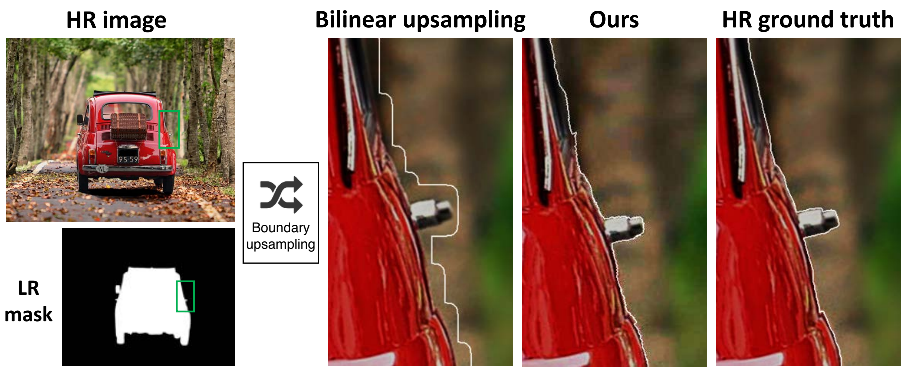


Figure 5.1: Concept overview. The example is from the newly created PixaHR dataset. Given low resolution mask and high resolution image on the left, a bilinear upsampling with scale factor 16× would results in boundary misalignment in high resolution image, as is shown in the enlarged boundary region on the right. Also, the new details in high resolution would be missed.

interaction [108, 112, 113, 117, 118]. While there has been some work on HR semantic segmentation [119,120] and upsampling [121,122], there is less focus on accurately capturing the boundary detail in HR. Instead of treating this problem as an upsampling problem, we treat it as boundary detection and harness the contents in HR images for prediction.

To this end, we propose a novel approach to handle boundary refinement in HR images. (See Figure 5.2.) Our key idea is to allow the power of deep learning methods to be applied to HR images in a time and memory efficient manner by operating on narrow images made up of pixels near the boundary. Given an accurate LR mask, the boundary in HR is likely in proximity to the upsampled LR boundary. (See Figure 5.1.) Therefore, to save memory and computation, we propose to search for the target boundary in a strip region near the boundary of the upsampled mask. The strip image is formed by sampling pixels along and normal to the upsampled mask boundary. Since the normals may not be smooth due to inaccurate boundaries

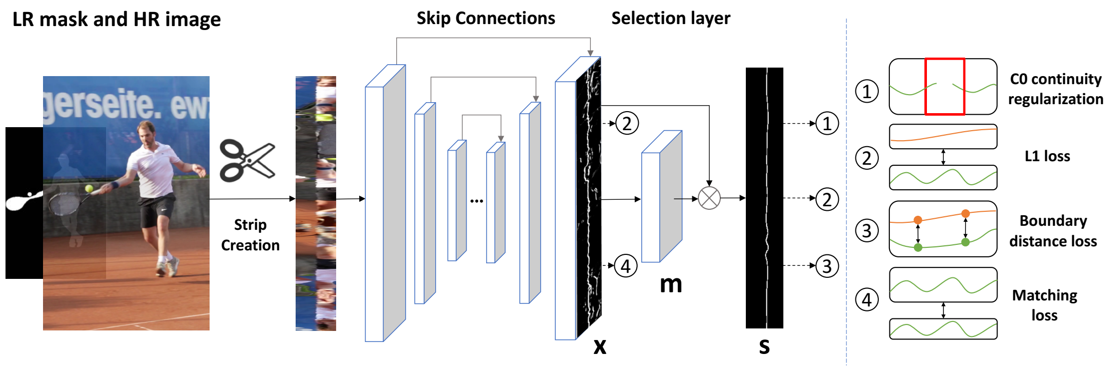


Figure 5.2: Framework. To save memory and computation, we predict the boundary in a strip image instead of the whole image. First, the strip image is extracted from the HR image and corresponding LR mask. Feeding the strip image as input, the network predicts all potential boundaries (denoted as "x") and passes the initial prediction to a selection layer (denoted as "m") to pursue more accurate prediction on the target boundary (denoted as "s"). The numbers are indicator to the losses displayed on the right. Orange and green curves denote the ground truth and prediction, respectively. Note that the strip image and prediction are rotated 90 degree for visualization.

in the upsampled mask, we represent the LR boundary with a spline approximation and directly treat the orthogonal derivatives of the upsampled spline as the normal directions. Feeding as input the generated strip images, we train a network to firstly predict all potential boundaries. Based on the initial prediction, an additional selection layer is included to predict the target boundary more accurately. To encourage closer prediction and reduce false positives, we propose loss functions to minimize the boundary distance between the prediction and ground truth in the strip image and to encourage C0 continuity in the prediction. Lastly, we pursue consistent results through matching the prediction under different strip sizes to further boost the performance.

To validate our approach, we create a new PixaHR dataset (see Figure 5.1 for image example) consisting of 100 photos with average resolution $7k \times 7k$ and evaluate our approach up to scale factor 32×. Results on DAVIS 2016 and COCO coarse annotations also show our ability to refine coarse boundary annotations.

In a nutshell, our contribution is three-fold. 1) We propose an approach to predict the boundary in a strip image which converts potential boundary regions into a strip space. This approach allows us to apply neural networks in a computationally and memory efficient manner. 2) To improve performance and encourage closer prediction, we propose novel losses including boundary distance, matching and C0 continuity loss. 3) We create a high resolution dataset for evaluation. To the best of our knowledge, we are the first learning based approach to make HR dense boundary refinement with resolution up to $10k \times 10k$. Extensive experiments on both public and the new PixaHR dataset strongly highlight our effectiveness.

5.2 Related Work

Boundary Refinement. Multiple attempts have been made to improve boundary quality through extracting better features [8,101,116,123,124]. Xie *et al.* [101] utilize features from multiple layers and fuse both low and high level features to detect edges. Liu *et al.* [116] explore rich convolutional features to boost the performance. More related, attention has been taken to refine coarse boundary predictions or annotations [8, 115]. Conventional methods like dense Conditional Random Fields (CRF) [125], Graph Cuts [126] model the relationship between nearby pixels and thus can be applied to refine LR masks [127]. However, these are segmentation based and only low-level features have been utilized. With more supervision, Yu *et al.* [115] propose to simultaneously learn and align edges to refine misaligned boundaries directly. Acuna *et al.* [8] further improve the performance by introducing a thinning layer and active alignment strategy to obtain refined boundary. These methods mainly explore edge detection in LR images. In contrast, we tackle HR boundary refinement and apply detection only on regions around upsampled LR boundary splines and thus is more memory and computation efficient.

Active Contours. Active contour models like Snakes [104] have been introduced to refine boundaries from coarse ones. Various approaches have been explored to handle the limitation of Snakes through, *e.g.*, better initialization, morphological operation [128] or user interaction [108]. Since our method also refines the curve upsampled from LR mask, we can benefit from these methods and refine the boundary further. Instead of taking the whole image as input, deep active contour [129] learns to predict the flow of boundary pixels in a patch by patch fashion. However, it cannot guarantee a continuous boundary prediction. Instead, our approach directly extracts a consecutive boundary region and thus contains more global information. Rather than predict the entire curve, other works have explored predicting control points [117,130,131] through recurrent neural networks or Graph Convolutional Networks (GCN) [132] and then fit a curve as the final prediction. However, boundary details are smoothed in the spline representation. In contrast, our approach predicts precise edge information directly. Another line of work implicitly represents boundary curves. For example, deep level set methods [133] evolve boundary curves by minimizing the level energy function. Other learning based approaches [7, 134, 135] have proposed to provide useful features, including texture, color or shape, for better optimization. However, these learning based approaches suffer from computation and memory issues when the resolution increases because they process the entire image while our approach only focuses on the regions around upsampled LR boundaries, and thus requires less computation and memory overhead.

High Resolution Up-sampling. With the information of low resolution masks, researchers have focused on achieving high quality HR segmentation masks. Conventional methods [136, 137] reach HR by applying upsampling jointly with the LR mask reference. However, the fixed filter structures have difficulty capturing new HR boundary details. He *et al.* [138] propose guided filtering to smooth while preserving edge information when upsampling. Wu *et al.* [121] make the guided filter faster and learnable. For HR segmentation approaches, Zhao *et al.* [120] propose to aggregate LR features for HR segmentation and Chen *et al.* [119] align both global and local features to avoid heavy GPU consumption for HR segmentation. Even though these methods can be potentially adapted to boundary refinement, our method mainly focuses on boundary regions and is designed to detect boundaries in HR directly. Therefore, our approach learns new HR boundaries better, especially when LR boundaries are coarsely annotated.

5.3 Approach

Our goal lies in refining boundaries in HR images given LR precise masks. To achieve this purpose efficiently, we propose to predict on a strip image that captures the potential boundary region rather than the entire HR image. Figure 5.2 illustrates

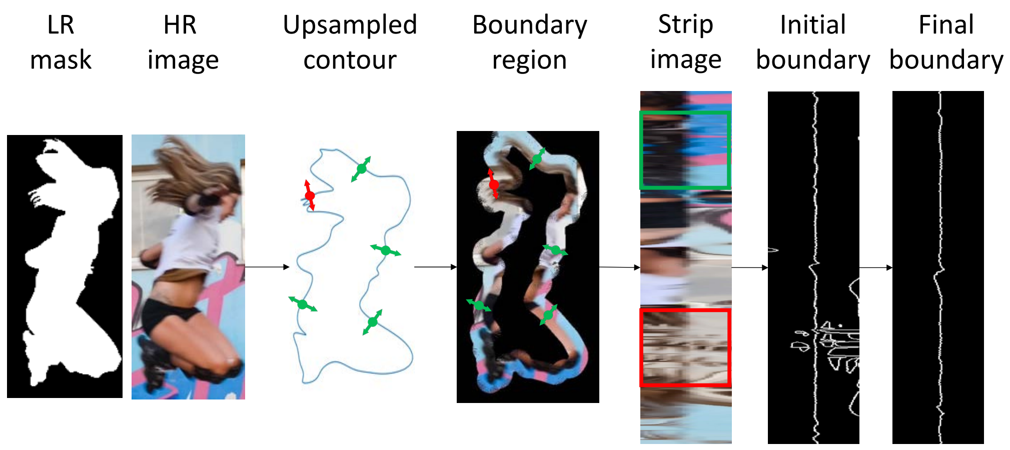


Figure 5.3: Strip image creation. To generate strip image, B-spline representation of the contour in the LR mask is upsampled to HR as a coarse boundary. The HR region along the normal direction (*e.g.*, red and green arrows) of the contour is then extracted. Finally, the strip image and corresponding boundary ground truth is obtained by flattening the extracted region in both the HR image and mask. Note that the final boundary filters out noisy boundaries (*e.g.*, the red box region) from the initial boundary. The strip image and boundaries are rotated 90 degree for visualization.

our framework. Our approach consists of strip image creation, which converts HR RGB image into strip image, strip boundary prediction, which refines the edges on the strip image using a network and strip reconstruction which reconstructs the prediction in the original image from the strip boundary prediction during testing.

5.3.1 Strip Image Creation

Figure 5.3 describes the procedure of strip image creation. Due to the interpolation introduced by upsampling, a directly upsampled boundary from the LR image is likely to be shifted from the ground truth boundary in HR. To localize the real HR boundary pixels, searching around the upsampled boundary is more necessary than searching the whole image. Therefore, we extract pixels near the upsampled boundary to create a strip image. To create the strip image, we step along the boundary and sample points along the normal direction at each point on the curve. To obtain smoothly varying normal directions along the coarse boundary, we represent the LR boundary by B-spline and upsample the LR spline to HR.

Given the HR image $I(p, q)$ and the upsampled spline representation $C = (p(k), q(k))$ of the boundary contour, where $(p(k), q(k))$ denotes the HR image coordinates parameterized by arclength k along the curve, the continuous strip image $J_{I,C}$ is defined by

$$J_{I,C}(k, t+H/2) = I(p(k) + t \times n_p(k), q(k) + t \times n_q(k)), \tag{5.1}$$

where t denotes the distance in the normal direction, H denotes the height of the strip image, and $(n_p(k), n_q(k))$ is the unit normal to the curve at arclength k. Accordingly, the strip image $J_{I,C}(j, i)$ with dimension $H \times W$ is obtained by sampling $k = j \times dk$, $t = i \times dt$, where tangential step size $dk = \lfloor |C|/W \rfloor$ and normal step size dt is set to 1 for simplicity. $|C|$ denotes the length of C, $j = 0, 1, ..., W$ and $i = -H/2, ..., 0, ..., H/2$. Also, bilinear interpolation is applied in the high resolution image to evaluate $I(p, q)$ for non-pixel coordinates (p, q).

The corresponding HR strip boundary ground truth is obtained similarly with two adaptations. First, for large sampling scale factors, the ground truth boundary is likely to be outside the range of the strip if the strip height is small, making the boundary in strip image not continuous. We add labels at the border of strip if no boundary pixel is included to maintain the C0 continuity of the boundary pixels in the strip image. Second, if the strip height is large, multiple boundary pixels might be included in each column in regions where the boundaries are closer than the strip height. In this case, we filter out the extraneous boundaries that are not connected to the current boundary. (See Figure 5.3.)

5.3.2 Strip Boundary Prediction

Provided the HR strip image as input, we train a network to predict the corresponding boundaries within the strip domain. For memory efficiency, we adapt light-weighted encoder-decoder based structure nested U-Net [139,140] for boundary prediction. Given the fact that proper dimension of strip image varies for different resolutions, we use instance normalization [73] during training so that the mean and variance are approximated per image.

As is shown in Figure 5.2, two prediction layers are proposed to learn the target boundary in strip image to account for the fact that multiple true boundaries may be present in a single column of the strip image. Firstly, we extract the last upsampling layer to predict all potential boundaries. This encourages the network to learn boundary features within the strip image. To predict the target boundary, we add a learnable selection layer to pick up the target boundary from potential boundaries. The input to the selection layer is the initial prediction, and we apply column-wise softmax to the output of the selection layer as a confidence score for the initial prediction. Finally, the target boundary is computed by the multiplication between the initial prediction and the selection score. The selection layer also smooths the initial prediction, analogous to the non-maximum suppression in Canny edge detection [100]. Formally,

$$s = x \odot m, \tag{5.2}$$

where $\odot$ denotes pixel-wise multiplication, s denotes the final prediction, x denotes the initial prediction which applies Sigmoid activation to the output of the last upsampling layer and m is the softmax activated output of the selection layer.

5.3.3 Loss Function

Our basic loss function for the initial and final boundary prediction is a weighted l_1 loss to differentiate the boundary from non-boundary pixels. Formally,

$$L_{\text{e}} = \beta \sum_{(i,j)\in Y_+} |y_{ij} - s_{ij}| + (1-\beta) \sum_{(i,j)\in Y_-} |y_{ij} - s_{ij}|, \tag{5.3}$$

where Y_+ and Y_- denote boundary and non-boundary pixels, respectively. $\beta = |Y_-|/|Y|$ denotes the weight to balance the label and $|Y|$ denotes the total number of pixels in strip mask. s_{ij} denotes the prediction and y_{ij} denotes the binary ground truth at position (i, j) in the strip image.

In addition, we adapt Dice loss [141] to boundary prediction to encourage intersection between prediction and ground truth:

$$L_{\text{dice}} = 1 - \frac{2\sum s_{ij} \times y_{ij} + \epsilon}{\sum s_{ij} + \sum y_{ij} + \epsilon}, \tag{5.4}$$

where ϵ denotes a small constant to avoid zero division. The loss aims to maximize the intersection over union between the prediction and ground truth.

5.3.3.1 Boundary Distance Loss

For boundary prediction, a closer prediction to the boundary ground truth is preferred. However, both weighted l_1 and dice loss are not sensitive to the distance from prediction to ground truth. Therefore, we introduce a boundary distance loss to measure the average distance between the predicted boundary and the ground truth to encourage closer prediction. Thanks to the strip domain which maps the regions along the normal direction in every column, the boundary distance can be calculated directly through the difference between the prediction and ground truth. Given the prior that only one boundary pixel exists in each column in the final strip mask, the boundary distance at every column can be measured by calculating the argmax difference at every column between the prediction and ground truth. Since argmax function is not differentiable, we approximate it through soft argmax before calculating the boundary distance and formulate the loss as

$$L_{\mathrm{d}} = \frac{1}{W} \sum_{j=1}^{W} |\operatorname*{softarg}_{i}(s_{ij}) - \arg\max_{i}(y_{ij})|, \tag{5.5}$$

where W is the width of strip mask and the soft argmax in each column (normal direction) is computed as

$$\operatorname*{softarg}_{i}(s_{ij}) = \sum_{i=1}^{H} \left(\frac{|s_{ij}|}{||S_j||_1} \times i \right), \tag{5.6}$$

where $||S_j||_1$ is the l_1 normalization of s_{ij} at column j. Since the final prediction s_{ij} encourages a unimodal distribution according to Equation 5.2, this loss enforces the column-wise maximum activation of the final prediction to match with that in ground truth.

5.3.3.2 Matching Loss

Since the strip height is fixed during training, to introduce variance and avoid overfitting on specific strip height, we augment the data through cropping the strip height. Starting from a large height, we crop the strip to a shorter one and make a new prediction. For consistency, the overlapped regions between original and the cropped strip should have the same initial prediction since all potential boundaries are predicted. Formally, we take a l_1 loss between the cropped and original initial prediction to calculate the matching loss,

$$L_{\mathrm{m}} = \frac{1}{|Y_{crop}|} \sum_{(i,j)\in Y_{crop}} |x'_{ij} - x_{ij}|, \tag{5.7}$$

where Y_{crop} is the cropped region of original mask Y and x'_{ij} is the new initial prediction for the cropped strip image. In addition, this loss also helps the network learn to ignore spurious edges detected near the border of the strip.

5.3.3.3 C0 Continuity Regularization

Additionally, we add a C0 continuity regularization to the final prediction to enforce a continuous prediction. Ideally, at most one boundary pixel is allowed at every column in the final prediction, so the prediction is C0 continuous if the maximum activated position of every column is C0 continuous. Specifically, we compute the soft argmax of every column, calculate a marginal difference between nearby argmax columns and penalize the position within a window size where prediction becomes discontinuous. Formally,

$$L_{\mathrm{C0}}{=}\frac{1}{W}\sum_{j=1}^{W}P(\max(0,|\underset{i}{\mathrm{softarg}}(s_{ij}){-}\underset{i}{\mathrm{softarg}}(s_{i,j+1})|{-}v)), \tag{5.8}$$

where v denotes the margin value and P denotes the maxpooling with a fixed kernel size so that all pixels within the range get penalized. s_{iW+1} is replicated by s_{i1} for calculation. This loss serves as a self regularization as no ground truth label is required.

The total loss function is therefore,

$$L_{\mathrm{total}} = L_e + L_{dice} + \lambda_1 L_d + \lambda_2 L_m + \lambda_3 L_{C0}, \tag{5.9}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyper-parameters to adjust the weight of each loss. L_e is applied to both the initial and final prediction. L_m is only applied to the initial prediction and L_{dice}, L_d, L_{C0} are applied only to the final prediction. With the total loss function, a closer prediction is preferred and the network draws attention to the target boundaries.

5.3.4 Strip Reconstruction

To make a prediction on the HR image, a mapping between the predicted strip boundaries and the full HR mask is required at inference. For every pixel in the strip image, the corresponding coordinates in the HR image are recorded for reconstruction. Given the raw prediction, we optimize the path with a dynamic programming similar to seam carving [142] and find the path with minimum energy. We minimize the function

$$E_{ij} = -s_{ij} - \frac{|\partial I(i,j)|}{max(|\partial I|)}, \tag{5.10}$$

where $|\partial I(i,j)|$ denotes the magnitude of the image gradients at (i,j). The algorithm searches for the energy cost for neighborhood pixels and finds the path with a minimum energy cost, which indicates the boundary path with the highest probability. We then connect the original coordinates of the final path in the full mask to form the full prediction.

At inference, the flexible input dimension of our framework enables different strip sizes for different images. Benefitting from it, we determine the width of strip, which reflects the number of sampling points along the boundary, by multiplying the LR boundary length with the scale factor. We fix the height of strip with the assumption that all target boundaries are involved, and an adaptive height adjustment strategy is also discussed in Section 5.4.6. For objects containing multiple contours due to complex topology, the prediction is made on each contour separately.

5.3.5 Implementation Details

We generate the spline curve efficiently from the binary mask using the scipy function 'splprep' after extracting contours. To guarantee a consistent sign for the normals, we extract strip images from closed contours. The starting point of strip is not deterministic so that no bias is introduced in training. The final ground truth strip boundary mask is obtained by taking the gradient of the ground truth segmentation mask after removing any isolated noisy boundaries. Additionally, we randomly add small shifts to the spline representation to introduce position variation of the target boundary in strip image during training. Our framework is implemented in Pytorch. The encoder consists of 4 3×3 convolutional layers and the decoder consists of 4 upsampling layers. The selection layer consists of another convolutional layer with 3×3 kernel size. The activation function is ReLU [143] for all encoder and decoder layers. We use instance normalization for all normalization layers to enable flexible input size at inference. During training, the input strip dimension is fixed as 80×4096. We train the network for 70 epochs with batch size 6 on an NVIDIA GeForce TITAN P6000. We use Stochastic Gradient Descent (SGD) as optimizer and the initial learning rate is 0.1. The learning rate decays by a factor of 10 after every 20 epochs. The momentum is set to 0.9 and weight decay is set to 0.0005. λ_1, λ_2 and λ_3 are set to be 0.1, 20 and 1 empirically. We crop strip image by half to obtain Y_{crop} for matching loss and the maxpooling kernel size for C0 continuity regularization is 11. The margin in C0 continuity regularization is set to 1. Horizontal flipping is applied as data augmentation.

Dataset	**DAVIS 2016** [56] 4×		**PixaHR** 8×		**PixaHR** 16×		**PixaHR** 32×	
Metrics	F(0 pix)	F(1 pix)	F(1 pix)	F(2 pix)	F(1 pix)	F(2 pix)	F(1 pix)	F(2 pix)
Bilinear Upsampling	0.171	0.521	0.116	0.194	0.15	0.187	0.07	0.106
Grabcut [103]	0.232	0.541	0.063	0.121	0.020	0.053	0.0	0.0
Dense CRF [125]	0.268	0.702	0.278	0.434	0.245	0.389	0.142	0.227
Bilateral Solver [137]	0.274	0.569	0.207	0.277	0.185	0.247	0.156	0.216
Curve-GCN [117]	0.076	0.160	0.021	0.033	0.018	0.028	0.012	0.028
DELSE [7]	0.271	0.531	0.096	0.133	0.086	0.132	0.080	0.130
STEAL [8]	0.171	0.348	0.282	0.457	0.151	0.255	0.09	0.144
JBU [136]	0.175	0.447	0.140	0.231	0.117	0.184	0.055	0.090
Guided Filtering [138]	0.129	0.349	0.121	0.195	0.092	0.145	0.060	0.097
Deep GF [121]	0.193	0.461	0.286	0.420	0.175	0.269	0.09	0.141
U-Net boundary	0.320	0.656	0.170	0.297	0.139	0.197	0.068	0.108
U-Net strip (baseline)	0.303	0.710	0.334	0.455	0.303	0.425	0.267	0.357
Ours	**0.423**	**0.788**	**0.416**	**0.508**	**0.396**	**0.498**	**0.330**	**0.447**

Table 5.1: Boundary-based F score comparison. The scale factor between low and high resolution image is 4 on DAVIS 2016 and 8, 16, 32 on PixaHR. For DAVIS 2016, the pixel dilation is 0 and 1 and for PixaHR is 1 and 2 instead.

5.4 Experiments

We evaluate our approach on two HR datasets which provide both low and high resolution ground truth in Section 5.4.2, and then analyze the importance of each components in our framework in Section 5.4.3. We also provide memory and speed comparison in Section 5.4.4.

5.4.1 Datasets and Metrics

For our experiments, we need a dataset with highly accurate pixel-level HR annotation. Unfortunately, most current datasets are low resolution and many provide inaccurate polygon boundaries as ground truth annotations. We found DAVIS [56] to provide accurate enough results with a resolution that is usable for our needs. To better evaluate the results at large scaling factors, we introduce a new dataset—PixaHR. We describe these datasets below.

DAVIS 2016 [56]: A benchmark for video segmentation which consists of 50 classes with precise annotations in both 480P and 1080P. To enlarge the scale factor, we down sample the 480P mask by a factor of 2, train our approach on the 30-class 1080P training set with 240P LR masks and test on 20-class 1080P testing set. The scale factor is 4.5 for this experiment. The results are evaluated frame by frame.

PixaHR: To evaluate more realistic scenarios, we create a PixaHR dataset. It contains 100 images with average resolution $7k \times 7k$ (ranging from $5k \times 5k$ to $10k \times 10k$) collected from public photograph website Pixabay [144]. We manually annotate the object boundary in the HR images, downsample the HR mask by $8\times$, $16\times$ and $32\times$ and obtain binary LR mask for evaluation. The photos were uploaded by public users and have diverse contents. We apply our model that was trained on DAVIS to this dataset for evaluation.

Metrics: We use boundary-based F score introduced by Perazzi *et al.* [56] for evaluation, which is designed to evaluate the boundary quality of segmentation. As it allows changing pixel tolerance by dilation, we set 0 and 1 pixel dilation on DAVIS, and 1 and 2 pixel on PixaHR dataset to measure how close the prediction is to the ground truth.

5.4.2 Main Results

For upsampling based approaches, we compare our approach with **Bilinear Upsampling**, **Bilateral Solver** [**137**], Joint Bilateral Upsampling [136] (**JBU**), **Guided Filtering** [**138**] and **Deep GF** [**121**]. The boundary is obtained by taking the gradient of the upsampled mask. For boundary refinement approaches, we compare with **Grabcut** [103], **Dense CRF** [125] and **STEAL** [8] using upsampled mask as initialization. For active contour methods, the baselines are **Curve-GCN** [117] and **DELSE** [7], and predictions on PixaHR are made in LR and upsampled to original resolution since the whole boundary region is required at inference. Learning based approaches are trained or fine-tuned on the training set of DAVIS and evaluated directly on all datasets. In addition, we also compare our own implemented baselines as below:

• **U-Net boundary**: We train U-Net directly on the full resolution images on DAVIS for boundary prediction. We concatenate both the full resolution image and upsampled masks as input so that the network learns to refine the coarse masks. The loss function is a weighted binary cross entropy following Xie *et al.* [101]. Similarly, we also add deep supervision and fuse all intermediate layers to obtain the final prediction. The prediction is made patch-by-patch with patch size 1920×1080 on PixaHR dataset.

• **U-Net strip (baseline)**: Our baseline method which learns to directly predict the target boundary on strip image. Only weighted $l1$ loss is used as loss function.

• **Ours**: Our full model which applies selection layer to predict the boundary in strip images with our boundary distance loss, matching loss and C0 continuity regularization.

Table 5.1 exhibits our advantage over the baselines. For the DAVIS dataset, a simple upsampling yields a boundary shift from the ground truth and thus performs poorly. Grabcut and dense CRF are segmentation based and thus yield worse performance than ours. Even though other methods including bilateral solver, JBU

Dataset	DAVIS 2016	PixaHR 16×
Metrics	F(0 pix)	F(1 pix)
U-Net strip	0.303	0.303
U-Net strip dice	0.323	0.320
U-Net strip dice + selection	0.372	0.328
U-Net strip dice + selection + BD	0.390	0.342
Our w/o matching	0.405	0.365
Ours	0.423	0.396

Table 5.2: Ablation analysis on two datasets. Each entry is the boundary-based F score tested on individual dataset.

and Deep GF leverage the low resolution mask, they are designed for general up-sampling instead of for boundary refinement and prediction. Curve-GCN fits the curve from the predicted control points which cannot generate as precise a boundary as ours. DELSE moves the contour along the gradient of its energy function, but is less robust than our approach which predicts the target boundary pixels. Additionally, our approach outperforms STEAL as the scale factor increases, indicating the active alignment in STEAL may not be accurate enough for pixel-level boundary prediction. Compared with U-Net boundary, predicting the boundary in strip image (U-Net strip) yields a slightly better performance, perhaps because the strip image narrows down the search space for target boundary. As expected, with our selection layer and proposed losses, we boost the performance further by better determining the target boundaries from other potential boundaries. A similar tendency is observed on PixaHR dataset. Note that in large scale factor 32, most of the methods fail to make close predictions to the ground truth while our method still has a relatively stable performance.

Methods	Memory (MB)	Speed (s/image)
Bilinear Upsampling	-	0.01/0.02
Grabcut [103]	-	5.17/320
Dense CRF [125]	-	3.22/310
Bilateral Solver [137]	-	4.18/158
JBU [136]	-	0.08/5.71
Guided filtering [138]	-	0.08/16.1
Deep GF [121]	-	0.07/3.95
STEAL [8]	7775/7959	43.1/4231
Curve-GCN [117]	17330/17330	0.93/75.2
DELSE [7]	17771/17771	1.02/20.4
U-net boundary	17000/17000	0.31/24.5
Ours	3300/3300	0.28/2.51

Table 5.3: Memory and speed comparison. Each entry is the memory or speed on DAVIS 2016/PixaHR dataset. We only compare the memory usage among learning-based approaches.

5.4.3 Ablation Analysis

We analyze the importance of each component in our framework as listed below:

- **U-Net strip dice**: Adding dice loss to the baseline.
- **U-Net strip dice + selection**: Adding dice loss and selection layer to the baseline.
- **U-Net strip dice + selection + BD**: Adding dice, boundary distance loss and selection layer to the baseline.
- **Ours w/o matching**: Adding additional C0 regularization. It is our full model without the matching loss.

Table 5.2 summarizes the comparison result. Starting from our baseline U-Net strip, adding dice loss encourages more intersection with the ground truth boundary and thus yields better performance. Comparing **U-Net strip + dice** with **U-Net**

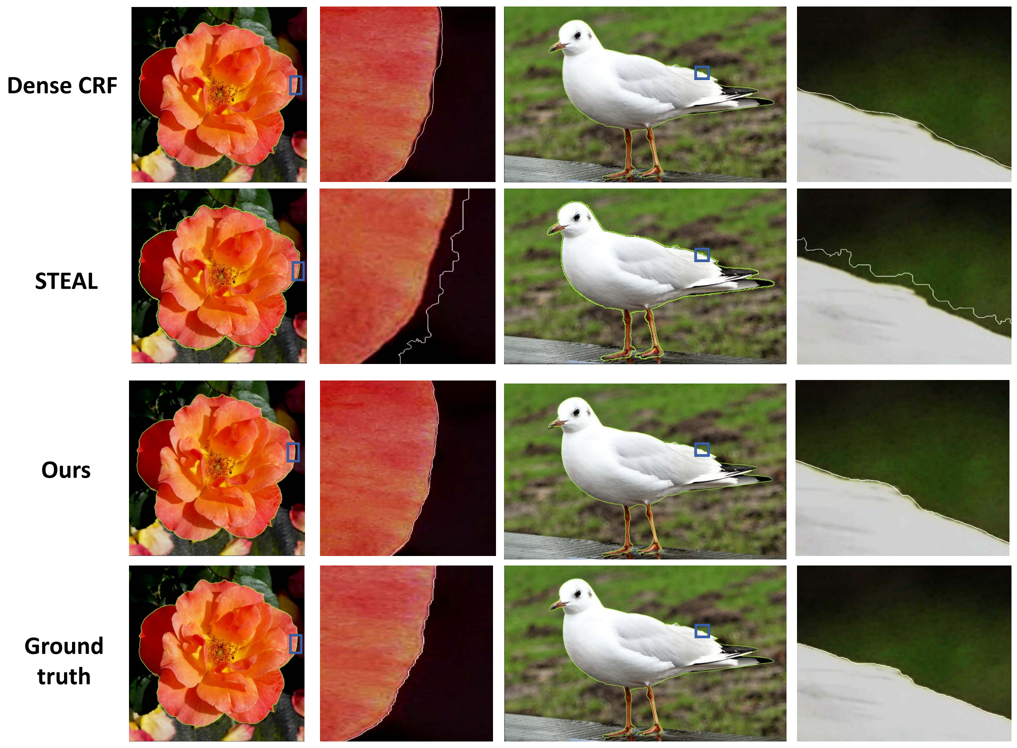


Figure 5.4: Qualitative results on PixaHR 32×. Rows from top to down are the results of Dense CRF, STEAL, Ours and the Ground truth. We show the entire boundary (green color) result first and enlarge the blue bounding box region for comparison (boundaries are whitened).

strip + dice + selection, the selection layer boosts the performance on DAVIS by a large margin, indicating it effectiveness in suppressing the noisy boundaries and smoothing the final prediction. Also, with the boundary distance loss the network learns to have closer prediction. With C0 regularization (**Ours w/o matching**), the network filters out false positive boundaries by making a continuous prediction. Finally, the performance further improves with the matching loss because the network makes a consistent prediction over different strip heights to avoid overfitting.

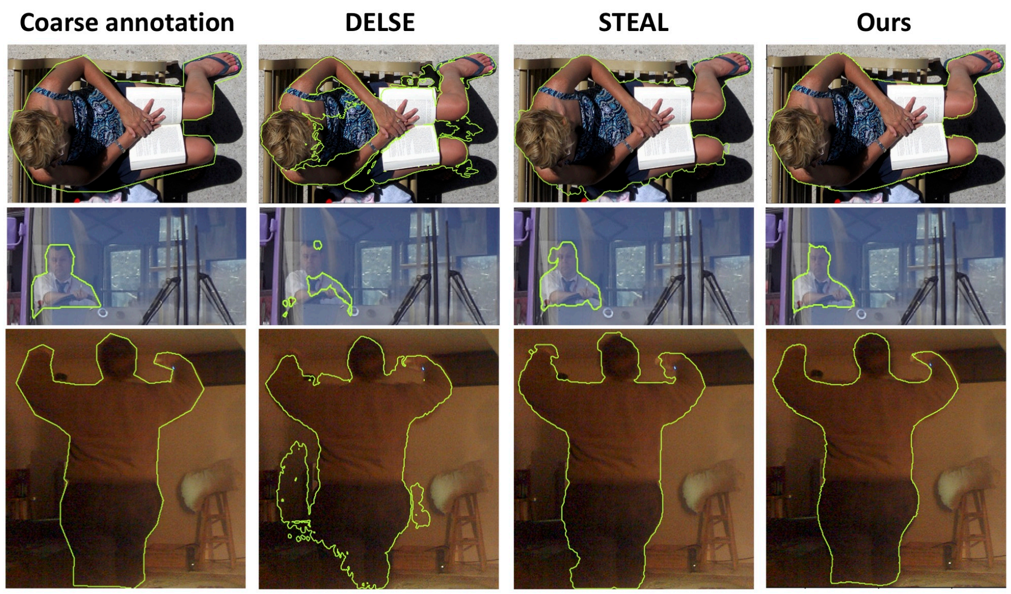


Figure 5.5: Qualitative results on COCO. Columns from left to right are coarse annotation, DELSE [7], STEAL [8] and Ours.

5.4.4 Memory and Speed Comparison

Since we only extract a strip image for prediction, our approach is efficient in both memory and computation. Table 5.3 compares our memory overhead and speed performance with baselines. Over all, our computation and memory requirement is relatively small. Our memory requirement is smaller than other learning based approaches. Note that for U-Net boundary and STEAL, the prediction on PixaHR is made patch-by-patch due to the high resolution.

More specifically, the main computation in our approach lies in strip reconstruction. *e.g.*, for a 1920×1080 DAVIS image with around 3200 pixels along the boundary, our strip image creation takes 0.08s, prediction process takes 0.06s and the strip reconstruction takes 0.14s. A similar computation percentage is observed

Dataset	PixaHR 32×
Metrics	F(1 pix)
Ours	0.330
Ours adaptive 1 segment	0.353
Ours adaptive 2 segments	0.365

Table 5.4: Strip height selection comparison on PixaHR 32×.

on PixaHR also.

5.4.5 Qualitative Results

We show visualization comparisons in Figure 5.4. It is clear that our approach produces more accurate boundariers than the other methods. To further show the effectiveness of our approach on refining the boundaries given LR or coarse masks, we provide qualitative results on COCO where only polygonal boundary ground truth is provided. We directly extract strip image using the coarse annotation on COCO, and visualize the prediction in Figure 5.5. Comparing with other approaches, our method provides more accurate boundaries, indicating the potential application of our approach to help refine the coarse boundaries.

5.4.6 Strip Height Adaptation

We predict the target boundary in the strip image under the assumption that the target boundary exists within the pre-defined height range, however, it might not hold true especially for a large scale factor. While one solution is to pre-define a larger height for strip image creation, we propose to progressively increase the height and regenerate strip image to make new predictions at inference. Specifically, we increase the height of strip image until the summation of the final prediction score decreases. Furthermore, height adjustment is more flexible by dividing the whole contour into several segments and adjusting them independently. The results are shown in Table 5.4. The comparison between **Ours** and **Ours adaptive 1 segment** indicates the effectiveness to have a flexible height. The performance increases further when dividing the whole contour into 2 segments which allows variable height for different regions.

5.5 Conclusion

In summary, this paper presents a novel strategy to handle HR boundary refinement computationally and memory efficiently given LR precise masks. To save memory, we propose to extract boundary regions along the upsampled boundary spline to form a strip image and make prediction within this strip image. To focus on the target boundaries in strip image, boundary distance, matching loss and C0 continuity regularization have been proposed. Extensive experiments on both public and our newly created dataset demonstrate the effectiveness of the proposed approach. However, the current approach still has difficulty predicting complicated topology and soft boundary regions. A smarter adaptive strip height adjustment for every pixel might be a potential solution, which is left for future research.

Chapter 6: Multi-model and Multi-level Knowledge Distillation for Incremental Learning

6.1 Introduction

Deep neural networks perform well on many visual recognition tasks [11, 145, 146] given specific training data. However, problem arises when adapting networks to unseen categories while remembering seen ones, which is known as catastrophic forgetting [147–149]. To tackle this issue, there is a growing research attention on incremental learning where the new training data is not provided upfront but added incrementally. The target of incremental learning is to achieve good performance on new data without sacrificing the performance on old and it has been widely explored across different tasks such as classification [9, 150] and detection [151].

To alleviate catastrophic forgetting in incremental learning, one possibility is to maintain a subset of old data to avoid over fitting on new data [9, 152, 153]. However, an issue in practice is that when models embedded in a product are delivered to customers, they no longer have access to trained data for privacy purposes. To tackle the situation, a stricter exemplar-free setting was introduced in [150], which requires no exemplar set for previous categories and only distills previous knowledge

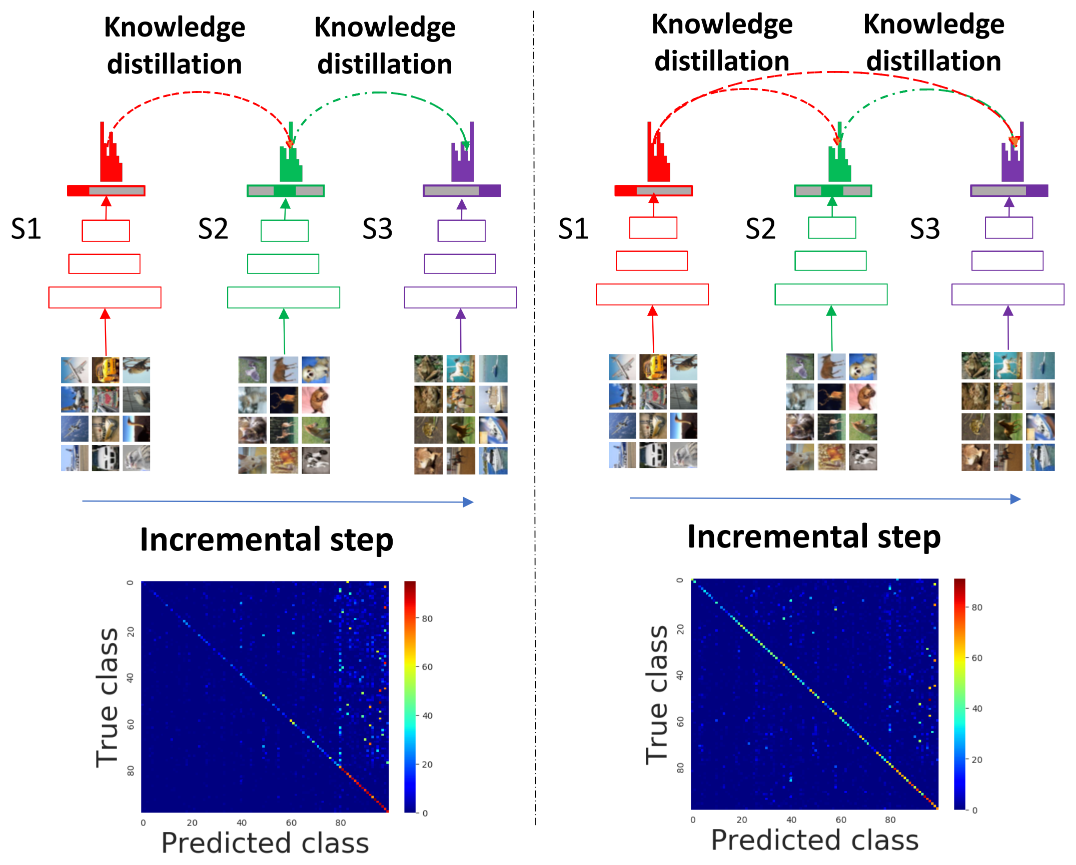


Figure 6.1: Concept overview. We propose to distill knowledge from all previous models efficiently to preserve old data information rather than sequentially applying distillation only to the last model. (For example, using both S1 and S2 in S3 for distillation instead of sequentially using S1 for S2 and then S2 for S3). The confusion matrix is LWF-MC [9] on the left and our method on the right for the exemplar-free incremental setting.

from the current categories.

Prior methods typically apply knowledge distillation [154] sequentially during the incremental procedure to preserve previous knowledge. Since they apply distillation only to the last model, it is difficult to maintain all past knowledge completely (the left side of Figure 6.1). From that observation, we propose using all the model snapshots. Prior knowledge is preserved better through our approach (the right side of Figure 6.1). However, saving all previous models may incur a great penalty in memory storage and without somehow compressing this historical information would not be practical. To address this, we reconstruct previous outputs using only "necessary" parameters during training.

To this end, we propose an end-to-end Multi-model and Multi-level Knowledge Distillation ($\boldsymbol{M}^2$KD) framework as depicted in Figure 6.2. We introduce a multi-model distillation loss which leverages the snapshots of all previous models to serve as teacher models during distillation, and then directly matches the outputs of a network with those from the corresponding teacher models. To make the pipeline more efficient, we adapt mask based pruning methods to reconstruct the previous models. We prune the network after each incremental training step and identify significant weights to reconstruct the model. This allows us to reconstruct previous models on-the-fly and utilize them as teacher models in our multi-model distillation. To further enhance the distillation process, we also include an auxiliary distillation loss to preserve more intermediate features of previous models. Additionally, our approach addresses catastrophic forgetting in sequential distillation, and thus generalizes well for both exemplar based and exemplar-free settings.

To show the effectiveness of our approach, we evaluate our model on Cifar-100 [155] and a subset of ImageNet [146]. We achieve state-of-the-art performance for all the datasets in exemplar-free setting. We also show improvement when adapting to exemplar-based incremental learning and our exemplar-free setting outperforms iCaRL [9] with a 200 exemplar budget.

In summary, our contributions are three fold. First, we propose a multi-model distillation loss, which directly matches logits of the current model with those from the corresponding teacher models. Secondly, for efficiency, we reconstruct historical models via mask based pruning such that model snapshots can be reconstructed with low memory footprint. Experiments on standard incremental learning benchmarks show that our method achieves state-of-the-art performance in exemplar-free incremental setting.

6.2 Related Work

The ultimate goal of incremental learning is to achieve good performance on new data while preserving the knowledge about old data. Generally, two types of evaluation settings [156] have been considered. One is multi-head incremental learning which utilizes multiple classifiers at inference, and the other is single-head incremental learning which only utilizes one classifier at inference.

Multi-head incremental learning. The evaluation setting in this stream is that a specific classifier is selected during testing according to the tasks or categories. With this prior information, no confusion exists across different classifiers, and thus the target becomes how to adapt the old model for new tasks or categories. Research has been focused on utilizing an episodic memory to trace back previous tasks [157–159], or constraining the important weights on old tasks [149]. In addition, [160, 161] learn a mask for pruning to further constrain the weights on old tasks. [162] distill the knowledge from the old model when adapting to new tasks. Different from this setting, we do not assume the task or category information is known during inference and follow the setting of single-head incremental learning. Also, even though we apply pruning in our approach, our goal is different from [160, 161] as the masks are utilized to reconstruct previous models and our approach requires no mask selection at inference.

Single-head incremental learning. Single-head evaluation uses only one classifier to predict both the old and the new classes. This setting is more challenging [156] compared to the multi-head counterpart because of the confusion between old and new categories. Knowledge distillation [154] is frequently utilized to preserve information. [150] distill the knowledge from the last model. [163] introduce Grad-CAM [164] in the loss function. A relaxed setting is to introduce exemplar set [9] for the old data and match previous logits through distillation. [152] explore the balance between old and new data during training. [153] focus on constructing exemplar set and [165, 166] replay the seen categories with GANs [83]. Instead of saving exemplars, we save the parameters of previous models for reconstruction. With that, this paper can be considered a complement research direction. In fact, as knowledge distillation is an important component in these methods, they can potentially benefit from our approach as well. Additionally, [167] alleviate the bias in knowledge distillation by introducing a scaling vector to trained classifier, however, our approach is agnostic to classifier and achieves better performance.

Network pruning. Considerable research has explored this area to reduce network redundancy. [168, 169] propose to compress network through quantization and Huffman coding. [170] compress the weights according to their scores. Other methods [171–173] explore compression for fast inference. In contrast to these methods, we leverage network redundancy and use pruning to reconstruct all previous models

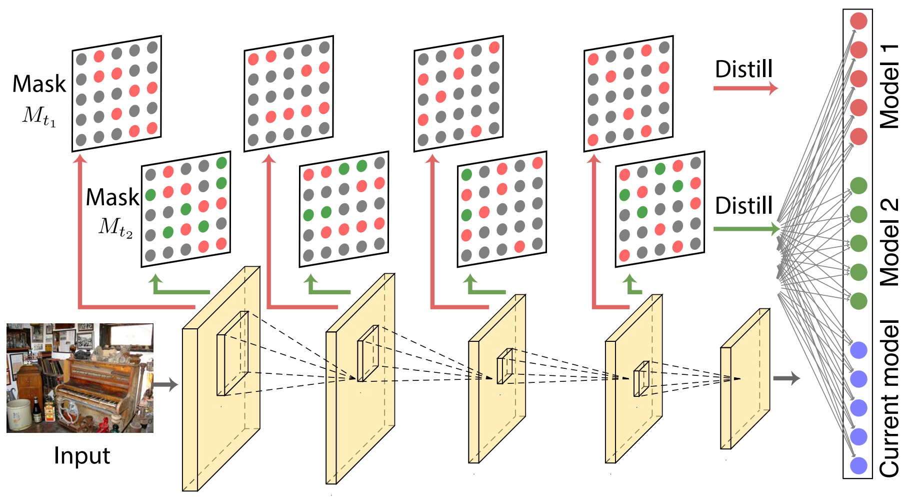


Figure 6.2: Framework overview. Given images from the current training data, we preserve previous knowledge directly from the reconstructed output through matching the logits with the corresponding model and classifying the current data with its ground truth. As an example, each layer contains a mask matrix M_{t_i} at the t_i-th incremental step recording significant weights for previous data. The gray dots represent the weights to be trained on the current data. The red and green dots are fixed during training, denoting the weights retained from the first and second incremental step respectively. The gray dots are fine-tuned for the current data before pruning. After pruning, a subset of the gray dots will be marked as important weights and become blue dots, and the remaining weights will be fine-tuned during the next incremental step. Accordingly, M_{t_2} is updated and used as M_{t_3} at the end of this round. In multi-model distillation, the red and green output logits of the current model are matched with the model 1 and 2 respectively while the blue logits are matched with its ground truth.

in incremental learning with low memory footprint.

6.3 Approach

We propose novel distillation losses to preserve previous information without introducing too much memory overhead (See Figure 6.2). The model is agnostic to the backbone architecture and generalizes well to both exemplar based and exemplar-free methods.

6.3.1 Multi-model Distillation

Single-head incremental learning consists of a sequence of incremental class inclusion process, referred to as incremental steps. Samples from a batch of new classes C_k are added at the k-th incremental step. For instance, 20 classes will be added per incremental step in a 20-class batch setting. Accordingly, the network assigns new logits (output nodes) for the incremental classes. At inference, the maximum logit score in the output is treated as the final decision.

The knowledge distillation used in incremental learning [9,150] mainly aims to match the output of the current model to a concatenation of the last model logits and ground truth labels. Formally, it optimizes the cross entropy for both the old and new logits,

$$\begin{aligned} L_D = &-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C_o} s_{ij}^{'} \log(s_{ij}) \\ &-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=C_o+1}^{C} y_{ij} \log(s_{ij}), \end{aligned} \tag{6.1}$$

where N and C denotes the number of samples and the total class number so far respectively, and C_o denotes the old classes. s_{ij} is the output score of the network obtained by applying Sigmoid function to the output logits for sample i at logit j. $s_{ij}^{'}$ denotes the old score obtained by the penultimate model. y_{ij} denotes the ground truth.

Treating the penultimate model as the teacher and applying this distillation sequentially helps preserve historical information, especially when no previous exemplar set is stored, which is the protocol for prior methods [9,150,152,163]. However, the historical information will be gradually lost in this sequential pipeline as the current model must reconstruct all the prior information from the penultimate model alone. To address this limitation, we propose multi-model distillation, which directly leverages all previous models as our teacher model set. Since we mainly have current training data and labels for both settings, the network is more confident on current classes than old ones. Therefore, matching the previous logits of the current model directly with their corresponding old models preserves information better than always using the last model. Formally, we minimize the cross entropy for the logits between the current model and corresponding teacher models from previous incremental steps,

$$\begin{aligned} L_{MMD} = &-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{P-1}\sum_{j=C_{k-1}+1}^{C_k} s_{ijk}^{'} \log(s_{ijk}) \\ &-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=C_{P-1}+1}^{C} y_{ij} \log(s_{ij}), \end{aligned} \tag{6.2}$$

where classes from $C_{k-1}+1$ to C_k belong to the k-th incremental step and P denotes the number of incremental steps. Classes from $C_{P-1}+1$ to C belong to the current categories. s_{ijk} is the output score of the current model for sample i at logit j in the k-th incremental step. $s_{ijk}^{'}$ denotes the output score of the k-th previous model.

Multi-model distillation matches the logits in the current model with the corre-

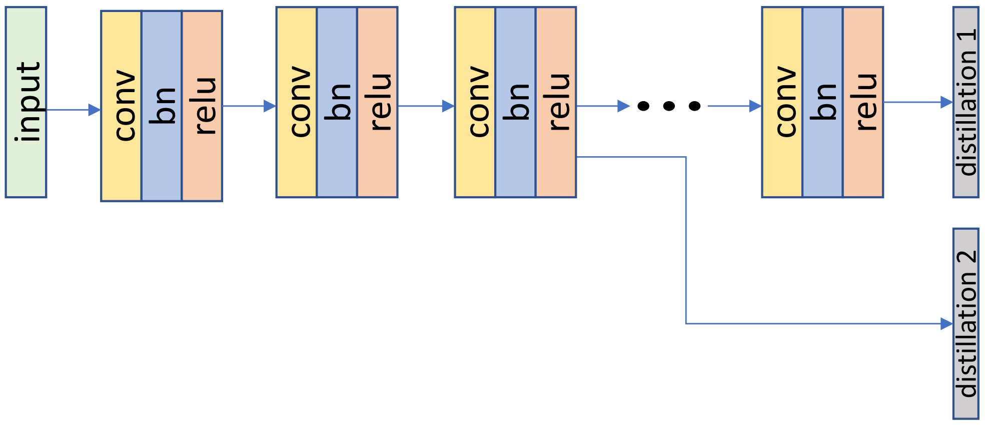


Figure 6.3: Illustration of auxiliary distillation. We extract the intermediate features and connect directly with an auxiliary classifier to preserve middle level knowledge.

sponding teacher model directly, reducing the information loss between incremental steps. At inference, we directly choose the maximum among the output logits, which acts as an ensemble of all the previous teacher models and the current model.

6.3.2 Auxiliary Distillation

Previous incremental learning methods preserve old class information through matching the final output. However, the features from intermediate layers also contain useful information. Inspired by the auxiliary loss in segmentation task [174], we propose an auxiliary distillation loss to preserve the intermediate statistics of previous models. Similar to using the final output to represent network statistics, the prediction made by lower level features also represents intermediate feature statistics. Following the main branch classification, we extract lower level features and use an auxiliary classifier to conduct classification based on intermediate features (See Figure 6.3).

Also, a multi-model distillation loss is added on this auxiliary classifier for the purpose of preserving prior lower level features, and a standard cross entropy loss is also included for classifying the current data. Formally,

$$\begin{aligned} L_{AD} = &-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{P-1}\sum_{j=C_{k-1}+1}^{C_k} a'_{ijk}\log(a_{ijk}) \\ &-\frac{\alpha}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(a_{ij}), \end{aligned} \tag{6.3}$$

where a'_{ijk} denotes the output score from previous auxiliary classifiers, a_{ijk} or a_{ij} is the output score of the auxiliary branch, α is the ratio between the distillation and cross entropy loss. Notice that all the logits in ground truth labels are utilized in the classification cross entropy to enforce the correct prediction of current data.

The total loss function of the network becomes,

$$L_{total} = L_{MMD} + \lambda L_{AD}, \tag{6.4}$$

where λ is the ratio between the main classification multi-model distillation and the auxiliary classification distillation. This auxiliary classification branch is only used during training. At inference time, we only use the main branch classifier for prediction.

Algorithm 1 Pruning Algorithm

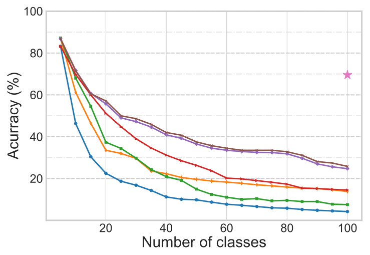
```
1: Input: $X^1, \ldots, X^k$    // input image sets of incremental step $1, \ldots, k$
2: $\Theta$            // current model parameters
3: store pre-update parameters and masks $m$
4: for $y = 1, \ldots, k$ do
5:     $Grad(\Theta_y(m < y)) = 0$    // apply mask
6:     update optimizer through Back-Propagation
         $\Theta_y \leftarrow min(L_{MMD}(\Theta_y) + \lambda L_{AD}(\Theta_y))$
7:     adjust threshold by pruning ratio    //update threshold
8:     $\Theta_y(\Theta_y < threshold) = 0$    // prune and update $\Theta_y$
9:     $m(\Theta_y >= threshold) = y$    //update masks
10: end for
```

6.3.3 Model Reconstruction

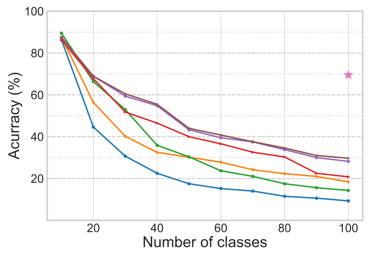
One drawback of multi-model distillation in its original form is that it utilizes all previous models, requiring additional memory storage for the models. However, we observe that distillation aims to match logits. Therefore it is only necessary to preserve the outputs of previous networks, not the entire networks themselves. Our idea is to save only a small set of the necessary parameters from which we can approximate the output. By that way, all the models can be recovered on-the-fly without large memory penalty.

To determine the necessary parameters, we adapt mask based pruning [160] for model reconstruction. Specifically, after training each incremental step we sort the magnitude of weights in each layer, freeze the important ones to reach a specified pruning ratio, and use the residual weights to train the next incremental class set. We repeat this procedure for all future incremental steps until all the incremental classes are included. (See Algorithm 1)

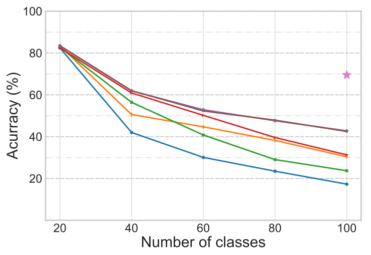
We use a mask M to identify the important weights of each layer for all previous incremental steps. After each pruning procedure, we update the mask for the



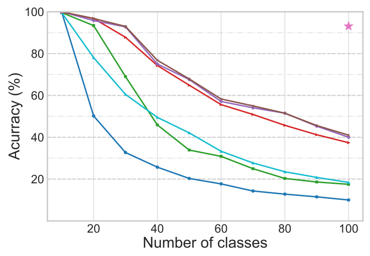
(a) Top-1 Cifar-100, 5-class batch



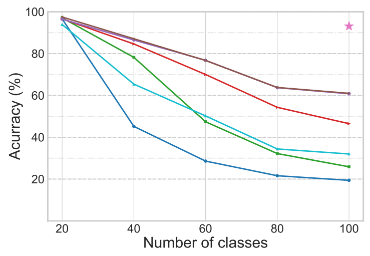
(b) Top-1 Cifar-100, 10-class batch



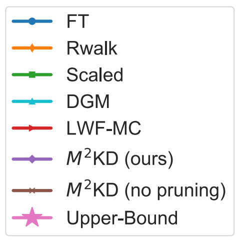
(c) Top-1 Cifar-100, 20-class batch



(d) Top-5 iILSVRC-small, 10-class batch



(e) Top-5 iILSVRC-small, 20-class batch



(f) Legend

Figure 6.4: Performance on iILSVRC-small and Cifar-100 dataset in exemplar-free setting. **(a)** Top-1 accuracy on Cifar-100 (5-class batch). **(b)** Top-1 accuracy on Cifar-100 (10-class batch). **(c)** Top-1 accuracy on Cifar-100 (20-class batch). **(d)** Top-5 accuracy on iILSVRC-small (10-class batch). **(e)** Top-5 accuracy on iILSVRC-small (20-class batch).

current incremental step. With the saved biases, batch normalization and classifier parameters, we can reconstruct all previous models from the last model (pre-updated model) on-the-fly. Formally, the output of a network with n convolutional layers is obtained from its classifier (the last layer) and features,

$$s = \Psi(f^{(n)}), \tag{6.5}$$

where Ψ denotes the classifier and $f^{(n)}$ denotes the features in the n-th layer and can be generally written as

$$f^{(n)} = \sigma(w^{(n)} f^{(n-1)} + b^{(n)}), \tag{6.6}$$

where w and b are weights and biases respectively, σ denotes the activation function and $f^{(0)}$ is the input.

With the mask M_k for the k-th incremental step, we reconstruct the corresponding features by:

$$f_k^{(n)} = \sigma(w_k^{(n)} \delta(M_k^{(n)} <= k) f_k^{(n-1)} + b_k^{(n)}), \tag{6.7}$$

where $M_k^{(n)}$ denotes the mask in the n-th layer at incremental step k, $f_k^{(n)}$ denotes the feature in the n-th layer in k-th incremental step, and δ denotes delta function.

Thus the output of the k-th model is reconstructed by

$$s_k = \Psi_k(f_k^{(n)}), \tag{6.8}$$

where s_k and Ψ_k denote the output of the network and the classifier for the k-th incremental step respectively.

6.4 Experiments

We first evaluate our method in the exemplar-free setting. Then we extend our method to the exemplar-based setting. For more analysis, we also compare our memory cost with other methods.

6.4.1 Datasets and Evaluation Metrics

The evaluation is conducted on iILSVRC-small [175] and Cifar-100 [155].

Evaluation Metrics. Following the same metrics in prior methods [9,150], the top-1 classification accuracy is reported for Cifar-100 and top-5 classification accuracy is reported for iILSVRC-small.

6.4.2 Exemplar-free setting

We evaluate our methods in exemplar-free single-head setting. For evaluation, we also compare with the following baselines and state-of-the-art single-head approaches.

FT: A baseline approach that only applies cross entropy loss to fine-tune the penultimate model on new coming incremental classes. Knowledge distillation is not applied.

Scaled [167]: A threshold moving strategy to alleviate the bias in knowledge distillation. We use the released code for evaluation.

DGM [166]: A dynamic generative memory approach which utilizes GANs to generate old samples as exemplar set. We use the released code for evaluation and no real sample is used during training.

Rwalk [156]: A generalization algorithm of EWC [149] and Path Integral [176]. The official code is evaluated.

LWF-MC [9]: A multi-class classification version of [150] as described in [9], applying distillation to the logits from the last previous model sequentially.

M^2KD (ours): Our full model applying multi-model, auxiliary distillation along with pruning to save memory storage.

Step	1	2	3	4	5
No pruning	**83.5**	**61.8**	**52.5**	**51.5**	42.1
Ratio 0.6	82.9	59.6	52.2	46.5	40.1
Ratio 0.7	83.5	61.7	52.5	50.0	**42.8**
Ratio 0.8	83.5	58.5	52.0	49.3	42.0
Ratio 0.9	83.0	58.0	49.7	47.3	39.9

Table 6.1: Top-1 accuracy comparison among different pruning ratios on Cifar-100 (20 classes per incremental step).

M^2KD (no pruning): The upper bound of our model which directly loads all the previous snapshots for multi-model distillation.

Upper-Bound: The upper bound of incremental learning which directly trains all classes together.

Figure 6.4 highlights our performance compared to state-of-the-art methods. For Cifar-100, our method consistently outperforms other methods from 5-class to 20-class batch per incremental step. The margin becomes larger as more incremental steps are added. This demonstrates the advantage of multi-model distillation as it avoids accumulating loss of historical information. Similar observation can be made when evaluating on iILSVRC-small. It is interesting to note that our model with pruning achieves comparable performance with the no-pruning version. This indicates the effectiveness of the pruning procedure in terms of saving memory while maintaining performance. Even though the residual active weights decrease gradually due to pruning, we still preserve the performance up to 20 incremental steps.

6.4.3 Ablation Studies

We investigate the effectiveness of each component of our method in this section. In particular, we compare our full model with the following baselines.

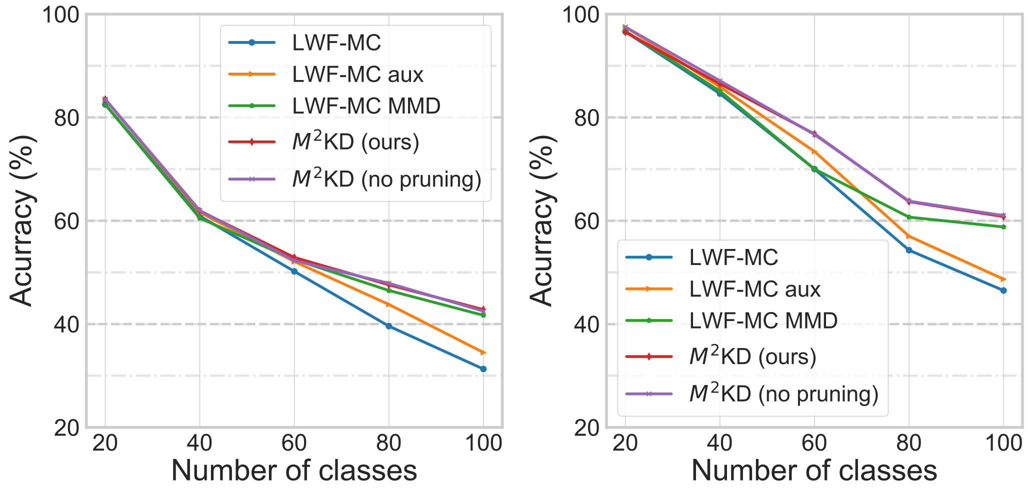
LWF-MC aux: Add auxiliary distillation to LWF-MC.

LWF-MC MMD: Change the original loss to our multi-model distillation. No auxiliary distillation is applied.

Ours skip1: Instead of using all previous models, we study the case when skipping some snapshots. Starting from the last previous model, we skip the first model in multi-model distillation. The skipped model is replaced by the next model for multi-model distillation.

Ours skip2: Skip the first two models instead of one compared to **Ours skip1**.

Figure 6.5 shows the comparison for each of the component in our approach. **LWF-MC aux** improves our baseline model **LWF-MC** on all the datasets after adding auxiliary distillation, indicating that the intermediate level information also contributes to preserving previous knowledge. With only multi-model distillation (**LWF-MC MMD**), the performance gradually improves for both datasets as more incremental steps are involved, which demonstrates that directly distilling knowledge from the corresponding model helps to reduce the lost in sequential distillation. Note that our multi-model distillation reduces to the standard distillation used in [150] if only one or two incremental steps are added. By incorporating the auxiliary distillation, however, our method still shows improved performance. Lastly, our model achieves nearly the same performance as our upper bound which saves all



(a) Top-1 Cifar-100 (b) Top-5 iILSVRC-small

Figure 6.5: Ablation Studies for our approach. **(a)** Top-1 accuracy comparison on Cifar-100 (20-class batch). **(b)** Top-5 accuracy performance on iILSVRC-small (20-class batch).

previous snapshots, showing the effectiveness of our pruning based approach.

Figure 6.6 compares how multi-model distillation is affected by the number of models. **LWF-MC** can be regarded as a special case which skips 3 models in the last round. The trend from **LWF-MC** to **Ours** shows that the performance improves as the number of model preserved increases, confirming the value of multi-model distillation.

6.4.4 Analysis on pruning ratio

We compare the results corresponding to different pruning ratios to investigate the robustness of our approach. Table 6.1 summarizes the results. Marginal performance variation (around 3%) is observed for different pruning ratios. Even though a higher (0.9) pruning ratio affects the performance as the active weights decrease in the current incremental step and a lower (0.6) ratio affects the performance as

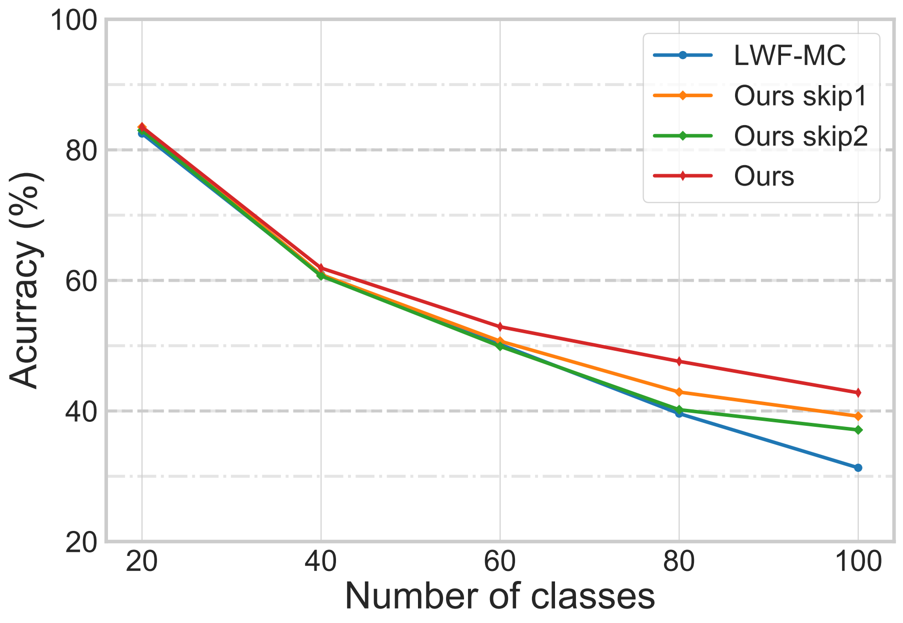


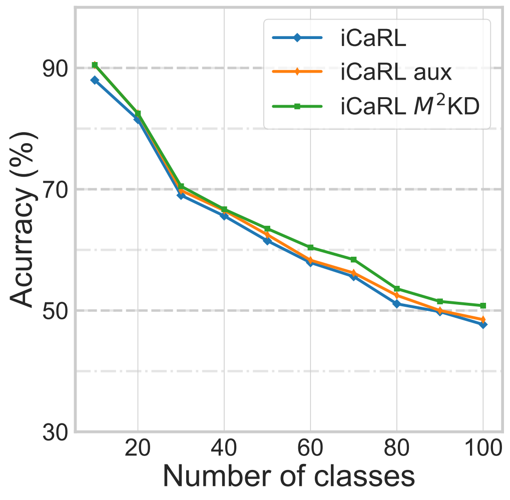
Figure 6.6: Comparison between different number of models used in multi-model distillation on Cifar-100 20-class batch.

available weights decrease in the future steps, the relatively trivial influence indicates that a large redundancy exists in the network architecture. Benefitting from it, our approach shows robustness to different pruning ratios.

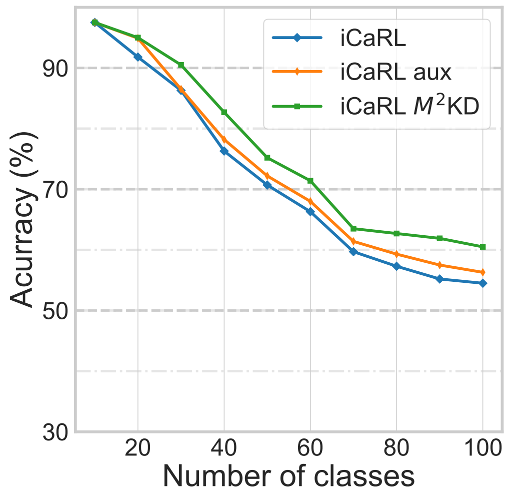
6.4.5 Exemplar Based Setting

Our approach can also be applied to exemplar based incremental learning methods which use distillation sequentially on the output of networks [9, 152, 153]. To evaluate our model in this setting, we add exemplar selection to our approach and compare with exemplar based methods.

iCaRL [9]: A prominent exemplar based incremental learning approach which constructs exemplar set for the old data according to the feature means and do distillation on the last previous model. A nearest class mean classifier [177] is applied at



(a) Top-1 Cifar-100



(b) Top-5 iILSVRC-small

Figure 6.7: Performance comparison in exemplar based setting. **(a)** Top-1 accuracy performance on Cifar-100 (10-class batch). **(b)** Top-5 accuracy performance on iILSVRC-small (10-class batch).

inference.

iCaRL aux: Adding auxiliary distillation to iCaRL.

iCaRL M^2KD: Change the original distillation function which only matches logits from the last previous model to our multi-model distillation. Auxiliary distillation is also appended for a better performance.

The results are shown in Figure 6.7. With the introduction of multi-model and auxiliary distillation, the performance of iCaRL improves. It indicates that with direct access to all the previous models for distillation, the knowledge preserves better even with exemplar set.

6.4.6 Memory Comparison

Starting from the memory footprint of LWF as our baseline, we compare the extra memory storage between exemplar based method such as iCaRL [9] and our approach. The memory is calculated in the 10-class incremental step setting for both iILSVRC-small and Cifar-100. For our approach, we directly calculate the storage difference between the last and the initial step. For iCaRL, the memory is approximately calculated by the average size of image for 2000 samples (*i.e.* the default exemplar size), and the compensation for saving the record of exemplar set. To optimize the memory consumption of iCaRL, we resize the images in iILSVRC-small to 256×256 and compress to JPG with quality 95 to match their network input size during training.

Table 6.2 shows the memory compensation for different methods. It indicates that our approach has approximately $7\times$ smaller memory compensation on iILSVRC-small and $10\times$ smaller on Cifar-100 than iCaRL. On average, for each incremental step, our approach only takes 0.98 MB and 0.08 MB for iILSVRC-small and Cifar-100 respectively. The memory advantage to exemplar based methods might become larger as higher resolution images take more storage.

We provide further memory analysis in Figure 6.8. We compare our approach with iCaRL on Cifar-100 given the same memory constraint. For fair comparison, we reduce the exemplar set as a penalty of the additional memory we use for network parameters to match with the memory size used for iCaRL. The performance is evaluated by averaging the top-1 accuracy across all the incremental steps. When memory budget equals to 200 images, we do not use any exemplar set but still perform better than iCaRL. The reason for this is that the sequential distillation pipeline tends to lose information even when exemplars from old classes are available. Moreover, increasing memory budget makes the performance gap between

Dataset	iILSVRC-small	Cifar-100
LWF-MC	0	0
iCaRL	68.0	9.4
$\boldsymbol{M}^2$KD (ours)	9.80	0.84

Table 6.2: Memory compensation comparison (MB). Each entry is the additional memory requirement for methods across different datasets based on the memory footprint of LWF.

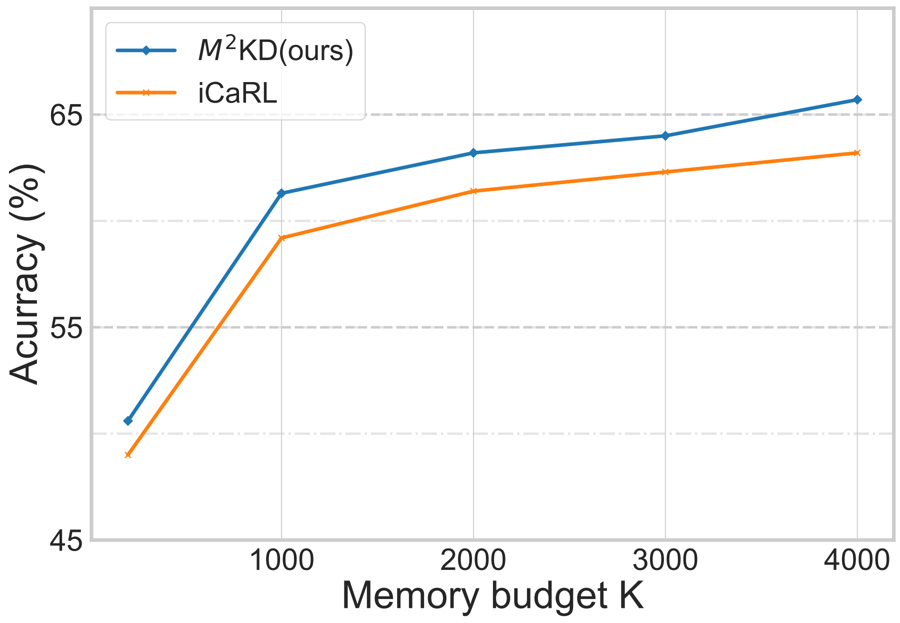


Figure 6.8: Analysis on performance and memory compared to iCaRL on Cifar-100 (10-class batch). We increase memory budget for exemplar set from 200 to 4000 images and report the average accuracy of all the 10 incremental steps.

our approach and iCaRL larger, showing our strength to memorize what has been learned.

6.5 Conclusion and Discussion

This paper presents a novel distillation strategy that mitigates catastrophic forgetting in single-head incremental learning setting. We introduce multi-model distillation which directly guides the model to distill knowledge from the corresponding teacher models. To further improve our performance, we incorporate auxiliary distillation to preserve intermediate features. More efficiently, we avoid saving all the model snapshots through reconstructing all previous models using mask based pruning algorithm. Extensive experiments on standard incremental learning benchmarks demonstrate the effectiveness of our approach. Incremental learning is still far from solved. A significant gap between one-step training versus incremental training still exists. It remains to be a open question how to reduce the confusion between different incremental steps especially without access to previous data, which might be a future exploration for our research.

Chapter 7: Conclusion

In this dissertation, we have studied the existing challenges in combining deep learning with forensics to make manipulation detection. We proposed RGB-N network to learn rich features to reveal more artifacts in the domain of local noise and RGB image. Moreover, we also extended from image manipulation to video manipulation detection and studied the problem of video inpainting detection. Furthermore, We combined a blending based GANs to improve the generalization of manipulation segmentation networks. We then studied the general issue with deep learning models. For the issue of high resolution prediction, we proposed a Deepstrip approach to handle inaccurate results at high resolution more efficiently. Lastly, we explored the field of incremental learning to prevent the catastrophic forgetting issue of current neural networks. Even though researchers have provided promising solutions to fight against the fake images/videos, the problem is still far from solved. Below we discuss some of the potential directions for the future research.

The first direction is to handle various manipulation techniques. We mainly focused on splicing and inpainting detection in the dissertation, however, detecting other manipulation techniques are also valuable. Taking into account this cat-and-mouse problem, the new emerging manipulation techniques including deepfake, generative model based image editing still remains to be explored. Applying deep learning to detect these new types of manipulation is an interesting direction for the future research.

Another challenge exists in manipulation detection is the domain shift problem. Research has demonstrated performance degradation when applying learned manipulation detection models to a different manipulation domain. This degradation is one of the major factors that limit the application of manipulation detection models. Exploring more generic features or discovering the domain specific to manipulation and applying domain generalization algorithms might be an interesting direction.

Bibliography

[1] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

[2] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016.

[3] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database. In *ChinaSIP*, 2013.

[4] Bihan Wen, Ye Zhu, Ramanathan Subramanian, Tian-Tsong Ng, Xuanjing Shen, and Stefan Winkler. Coverage—a novel database for copy-move forgery detection. In *ICIP*, 2016.

[5] Tiago José De Carvalho, Christian Riess, Elli Angelopoulou, Helio Pedrini, and Anderson de Rezende Rocha. Exposing digital image forgeries by illumination color classification. *TIFS*, 2013.

[6] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *ECCV*, 2018.

[7] Zian Wang, David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Object instance annotation with deep extreme level set evolution. In *CVPR*, 2019.

[8] David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *CVPR*, 2019.

[9] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[11] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[12] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016.

[13] Ruichi Yu, Xi Chen, Vlad I Morariu, and Larry S Davis. The role of context selection in object detection. In *BMVC*, 2016.

[14] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *CVPR*, 2018.

[15] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep grabcut for object selection. *arXiv preprint arXiv:1707.00243*, 2017.

[16] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Two-stream neural networks for tampered face detection. In *CVPRW*, 2017.

[17] Xunyu Pan, Xing Zhang, and Siwei Lyu. Exposing image splicing with inconsistent local noise variances. In *ICCP*, 2012.

[18] Miroslav Goljan and Jessica Fridrich. Cfa-aware features for steganalysis of color images. In *SPIE/IS&T Electronic Imaging*, 2015.

[19] Davide Cozzolino and Luisa Verdoliva. Single-image splicing localization through autoencoder-based anomaly detection. In *WIFS*, 2016.

[20] Davide Cozzolino, Diego Gragnaniello, and Luisa Verdoliva. Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques. In *ICIP*, 2014.

[21] Jawadul H Bappy, Amit K Roy-Chowdhury, Jason Bunk, Lakshmanan Nataraj, and BS Manjunath. Exploiting spatial structure for localizing manipulated image regions. In *ICCV*, 2017.

[22] Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Improved dct coefficient analysis for forgery localization in jpeg images. In *ICASSP*, 2011.

[23] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *TIFS*, 2012.

[24] Tian-Tsong Ng, Jessie Hsu, and Shih-Fu Chang. Columbia image splicing detection evaluation dataset. http://www.ee.columbia.edu/ln/dvmm/downloads/authspliceddataset/authspliceddataset.htm, 2009.

[25] Nist nimble 2016 datasets. https://www.nist.gov/itl/iad/mig/nimble-challenge-2017-evaluation/.

[26] Jing Dong, Wei Wang, and Tieniu Tan. Casia image tampering detection evaluation database 2010. http://forensics.idealtest.org.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[28] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Splicebuster: A new blind image splicing detector. In *WIFS*, 2015.

[29] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *IH&MMSec*, 2017.

[30] Yuan Rao and Jiangqun Ni. A deep learning approach to detection of splicing and copy-move forgeries in images. In *WIFS*, 2016.

[31] Jiansheng Chen, Xiangui Kang, Ye Liu, and Z Jane Wang. Median filtering forensics based on convolutional neural networks. *Signal Processing Letters*, 2015.

[32] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *IH&MMSec*, 2016.

[33] Ying Zhang, Jonathan Goh, Lei Lei Win, and Vrizlynn LL Thing. Image region forgery detection: A deep learning approach. In *SG-CRC*, 2016.

[34] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). *arXiv preprint arXiv:1709.02016*, 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[36] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. Large-scale evaluation of splicing localization algorithms for web images. *Multimedia Tools and Applications*, 2017.

[37] Neal Krawetz. A picture's worth... *Hacker Factor Solutions*, 2007.

[38] Babak Mahdian and Stanislav Saic. Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 2009.

[39] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. *TIFS*, 2012.

[40] Sungho Lee, Seoung Wug Oh, DaeYeun Won, and Seon Joo Kim. Copy-and-paste networks for deep video inpainting. In *ICCV*, 2019.

[41] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Deep video inpainting. In *CVPR*, 2019.

[42] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. Deep flow-guided video inpainting. In *CVPR*, 2019.

[43] Seoung Wug Oh, Sungho Lee, Joon-Young Lee, and Seon Joo Kim. Onion-peel networks for deep video completion. In *ICCV*, 2019.

[44] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. Free-form video inpainting with 3d gated convolution and temporal patchgan. *ICCV*, 2019.

[45] Jia-Bin Huang, Sing Bing Kang, Narendra Ahuja, and Johannes Kopf. Temporally coherent completion of dynamic video. *TOG*, 2016.

[46] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, 2018.

[47] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *CVPR*, 2019.

[48] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[49] Peng Zhou, Xintong Han, Vlad I Morariu, and Larry S Davis. Learning rich features for image manipulation detection. In *CVPR*, 2018.

[50] Yue Wu, Wael AbdAlmageed, and Premkumar Natarajan. Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In *CVPR*, 2019.

[51] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.

[52] Haodong Li and Jiwu Huang. Localization of deep inpainting using high-pass fully convolutional network. In *ICCV*, 2019.

[53] Qiong Wu, Shao-Jie Sun, Wei Zhu, Guo-Hui Li, and Dan Tu. Detection of digital doctoring in exemplar-based inpainted images. In *ICMLC*, 2008.

[54] Wei Wang, Jing Dong, and Tieniu Tan. Tampered region localization of digital color images based on jpeg compression noise. In *IWDW*, 2010.

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[56] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

[57] Kaiming He and Jian Sun. Image completion approaches using the statistics of similar patches. *TPAMI*, 2014.

[58] James Hays and Alexei A Efros. Scene completion using millions of photographs. *TOG*, 2007.

[59] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ToG*, 2017.

[60] Yunqiang Liu and Vicent Caselles. Exemplar-based image inpainting using multiscale graph cuts. *TIP*, 2012.

[61] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

[62] Haotian Zhang, Long Mai, Ning Xu, Zhaowen Wang, John Collomosse, and Hailin Jin. An internal learning approach to video inpainting. In *ICCV*, 2019.

[63] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ToG*, 2009.

[64] Chuan Wang, Haibin Huang, Xiaoguang Han, and Jue Wang. Video inpainting by jointly learning temporal structure and spatial details. In *AAAI*, 2019.

[65] Pasquale Ferrara, Tiziano Bianchi, Alessia De Rosa, and Alessandro Piva. Image forgery localization via fine-grained analysis of cfa artifacts. In *TIFS*, 2012.

[66] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. Detecting image splicing in the wild (web). In *ICMEW*, 2015.

[67] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). In *JVCI*, 2018.

[68] Peng Zhou, Bor-Chun Chen, Xintong Han, Mahyar Najibi, Abhinav Shrivastava, Ser Nam Lim, and Larry S Davis. Generate, segment and refine: Towards generic manipulation segmentation. *AAAI*, 2020.

[69] Xinshan Zhu, Yongjun Qian, Xianfeng Zhao, Biao Sun, and Ya Sun. A deep learning approach to patch-based image inpainting forensics. *SPIC*, 2018.

[70] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. In *TPAMI*, 2018.

[71] Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *ECCV*, 2016.

[72] Mengye Ren and Richard S Zemel. End-to-end instance segmentation with recurrent attention. In *CVPR*, 2017.

[73] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, 2016.

[74] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[75] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[77] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. In *JMLR*, 2014.

[78] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[79] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[80] Raymond A Yeh, Chen Chen, Teck-Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, 2017.

[81] Jinseok Park, Donghyeon Cho, Wonhyuk Ahn, and Heung-Kyu Lee. Double jpeg detection in mixed jpeg quality factors using deep convolutional neural network. In *ECCV*, 2018.

[82] Ronald Salloum, Yuzhuo Ren, and C-C Jay Kuo. Image splicing localization using a multi-task fully convolutional network (mfcn). In *JVCI*, 2018.

[83] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[84] Daniel Moreira, Aparna Bharati, Joel Brogan, Allan Pinto, Michael Parowski, Kevin W Bowyer, Patrick J Flynn, Anderson Rocha, and Walter J Scheirer. Image provenance analysis at scale. *arXiv preprint arXiv:1801.06510*, 2018.

[85] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[86] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

[87] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. In *ICCV*, 2017.

[88] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[89] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization. In *CVPR*, 2017.

[90] Jean-Francois Lalonde and Alexei A Efros. Using color compatibility for assessing image realism. In *ICCV*, 2007.

[91] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[92] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[93] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *CVPR*, 2017.

[94] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *CVPR*, 2017.

[95] Hieu Le, Tomas F Yago Vicente, Vu Nguyen, Minh Hoai, and Dimitris Samaras. A+ d net: Training a shadow detector with adversarial shadow attenuation. In *ECCV*, 2018.

[96] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *TOG*, 2003.

[97] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[98] Yizong Cheng. Mean shift, mode seeking, and clustering. In *TPAMI*, 1995.

[99] Andreas Opelt, Axel Pinz, and Andrew Zisserman. A boundary-fragment-model for object detection. In *ECCV*, 2006.

[100] John Canny. A computational approach to edge detection. *TPAMI*, 1986.

[101] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *CVPR*, 2015.

[102] Meng Tang, Lena Gorelick, Olga Veksler, and Yuri Boykov. Grabcut in one cut. In *ICCV*, 2013.

[103] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *TOG*, 2004.

[104] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *IJCV*, 1988.

[105] Tiantian Wang, Lihe Zhang, Shuo Wang, Huchuan Lu, Gang Yang, Xiang Ruan, and Ali Borji. Detect globally, refine locally: A novel approach to saliency detection. In *CVPR*, 2018.

[106] Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *CVPR*, 2019.

[107] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *CVPR*, 2019.

[108] Hoang Le, Long Mai, Brian Price, Scott Cohen, Hailin Jin, and Feng Liu. Interactive boundary prediction for object selection. In *ECCV*, 2018.

[109] Hongyu Xu, Xutao Lv, Xiaoyu Wang, Zhou Ren, Navaneeth Bodla, and Rama Chellappa. Deep regionlets for object detection. In *ECCV*, 2018.

[110] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[111] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[112] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018.

[113] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019.

[114] Hexiang Hu, Shiyi Lan, Yuning Jiang, Zhimin Cao, and Fei Sha. Fastmask: Segment multi-scale object candidates in one shot. In *CVPR*, 2017.

[115] Zhiding Yu, Weiyang Liu, Yang Zou, Chen Feng, Srikumar Ramalingam, BVK Vijaya Kumar, and Jan Kautz. Simultaneous edge alignment and learning. In *ECCV*, 2018.

[116] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *CVPR*, 2017.

[117] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019.

[118] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, 2016.

[119] Wuyang Chen, Ziyu Jiang, Zhangyang Wang, Kexin Cui, and Xiaoning Qian. Collaborative global-local networks for memory-efficient segmentation of ultra-high resolution images. In *CVPR*, 2019.

[120] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.

[121] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Fast end-to-end trainable guided filter. In *CVPR*, 2018.

[122] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018.

[123] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. In *CVPR*, 2016.

[124] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *ECCV*, 2018.

[125] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NeurIPS*, 2011.

[126] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[127] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ToG*, 2004.

[128] Luis Álvarez, Luis Baumela, Pedro Henríquez, and Pablo Márquez-Neila. Morphological snakes. In *CVPR*, 2010.

[129] Christian Rupprecht, Elizabeth Huaroc, Maximilian Baust, and Nassir Navab. Deep active contours. *arXiv preprint arXiv:1607.05074*, 2016.

[130] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[131] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018.

[132] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[133] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *JCP*, 1988.

[134] Diego Marcos, Devis Tuia, Benjamin Kellenberger, Lisa Zhang, Min Bai, Renjie Liao, and Raquel Urtasun. Learning deep structured active contours end-to-end. In *CVPR*, 2018.

[135] Dominic Cheng, Renjie Liao, Sanja Fidler, and Raquel Urtasun. Darnet: Deep active ray network for building segmentation. In *CVPR*, 2019.

[136] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. In *ToG*, 2007.

[137] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *ECCV*, 2016.

[138] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *TPAMI*, 2012.

[139] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[140] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *DLMIA*, 2018.

[141] Ken CL Wong, Mehdi Moradi, Hui Tang, and Tanveer Syeda-Mahmood. 3d segmentation with exponential logarithmic loss for highly unbalanced object sizes. In *MICCAI*, 2018.

[142] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *TOG*, 2007.

[143] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[144] Pixabay. `https://pixabay.com`.

[145] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[146] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[147] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, 1989.

[148] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *ICLR*, 2014.

[149] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

[150] Zhizhong Li and Derek Hoiem. Learning without forgetting. *TPAMI*, 2018.

[151] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *CVPR*, 2017.

[152] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018.

[153] Yu Li, Zhongxiao Li, Lizhong Ding, Peng Yang, Yuhui Hu, Wei Chen, and Xin Gao. Supportnet: solving catastrophic forgetting in class incremental learning with support data. *arXiv preprint arXiv:1806.02942*, 2018.

[154] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[155] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Tech.rep.*, 2009.

[156] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*, 2018.

[157] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

[158] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

[159] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

[160] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018.

[161] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 2018.

[162] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *ECCV*, 2018.

[163] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. *arXiv preprint arXiv:1811.08051*, 2018.

[164] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[165] Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Continual state representation learning for reinforcement learning using generative replay. *NeurIPS*, 2018.

[166] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. Learning to remember: A synaptic plasticity driven framework for continual learning. In *CVPR*, 2019.

[167] Khurram Javed and Faisal Shafait. Revisiting distillation and incremental classifier learning. In *ACCV*, 2018.

[168] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *ICLR*, 2016.

[169] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

[170] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *CVPR*, 2018.

[171] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *ICLR*, 2016.

[172] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *BMVC*, 2014.

[173] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[174] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[175] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[176] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017.

[177] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, 2012.