Shallow- and Deep- fake Image Manipulation Localization Using Deep Learning

Abstract—Forged image localization is an important research task, as such images may have a tremendous impact of various aspects of society. Images can be manipulated using image editing tools (known as “shallowfakes”) or, recently, artificial intelligence techniques (“deepfakes”). While there are many existing works that are designed for manipulated areas localization on either shallow- or deep-fake images, there is no single solution that works for both cases. In this paper, we propose the first solution that can perform the localization task on both shallow- and deep-fake images, with high inference accuracy. The dataset and code are available at:

https://github.com/jyhedHR/TSYP_CS_CHALLENGE_EspritSB_GRINTA.git

Keywords—Image manipulation, Manipulation localization, shallowfakes, deepfakes

I.INTRODUCTION

Images are a pivotal point in our daily life especially through social media. They are prone to many possible aberrations since it's relatively easy to swap or modify people's identities, posing a significant threat to our society due to their ability to influence people, and that's done either by shallowfakes (simply done by software editing tools)or deepfakes (done by ai tools) which could potentially be causing harm to reputations and influencing public opinion. Shallowfake detection methods are no longer used to determine the authenticity of an image but it goes more in depth to localize the forged regions in the image. This is done by generating a mask that represents the exact parts of the image that have been manipulated. One such method is MVSS-Net, which considers features from both the image pixels and the noise distribution extracted by BayarConv. It also identifies edges between real and fake areas to generate more accurate localization masks. After training the mvss_net model, we can observe performances that are not very impressive but help reveal intresting data that shows better performances when it comes to ShallowFake being compared to DeepFake. When it comes to ShallowFakes, the model provides an accuracy level that reaches up to 67%, this higher accuracy is due to it being way less sohpisticated and much more simpler manipulations meanwhile it*Network Design* gives an even less impressive 47% accuracy in DeepFake situations and that's due to the nature and complexity of these frame manipulations since it uses much more advanced techniques that make it even harder to spot the alternations. Modifcations that are used for backgrounds but pixels that might concern manipualted face regions are the main reason to the drop in the accuracy thanks to the techniques used in that latter method. In this paper, we'd like to propose a global solution that will be a key to localize and proove the existince of any type of image manipulation whether it was a Shallow or Deep fake, or both at the same time, and the fact that there is no actual opproach to resolve this problem today makes our proposal of very high importance. 1.we created a new dataset full of both deep- and shallow-fake frames to train the network 2.nzid nasaal e rojla The overall evaluation of our performances show that our model can achieve precise and accurate accuracy levels on both types of modifications, individually

II.PROPOSED METHODOLOGY

A.Network Design

According to the precedent scientific paper , researchers built their network design upon the UPerNet , The UPerNet is a multi-task framework developed to learn from heterogeneous image annotations in the CNN architecture for semantic segmentation.

It is designed to improve the accuracy of segmentation results by optimizing the rough segmentation results using a boundary optimization module . The UPerNet architecture combines the strengths of both the encoder-decoder architecture and the spatial path[1] with lateral connections.

To better understand their work , we started by breaking down each step:

1.Extracting Noise Distribution:To capture color information efficiently and informatively, they used BayerConv, which leverages Bayer filters (commonly found in image sensors) alongside a set of learnable high-pass filters. These high-pass filters analyze the image and extract both its colors and the distribution of noise, which often varies between authentic and manipulated regions, providing valuable clues for detecting forgeries.

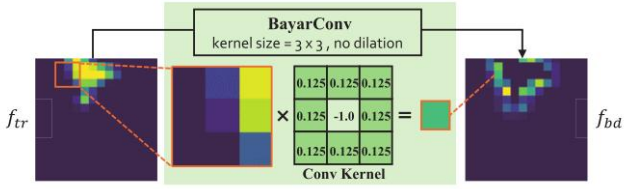

Fig1 :Illustration of the transformation process

3. **Feature Extraction:**The network utilizes two ResNet50 backbones to extract multi-level feature representations from the input image. These backbones are known for their effective feature extraction capabilities, capturing both low-level details and high-level semantic information.

3.**Feature Fusion**:Feature maps generated from each ResNet50 at each level are concatenated layer-wise, combining information across different scales. This allows the network to learn richer representations of the image content.ResNet-50 utilizes residual blocks that are key building of the networks consisting of the following 5 block starting by "Identity Mapping" ensuring that informations flow through every layer without vanishing, each residual block uses identity shortcut connection adding the input and output directly, and it's followed by "Residual Mapping" that plays its role by linking each block and performs its main feature exctraction in which we find their output being added to the input via aformentioned "Skip connection" that allows the network to retain valuable information from precedent layers and progressing through deeper ones which is pivotal for learning complex features and optimising the overall accuracy, being the powerhouse of ResNet, it also facilitates the flow of gradients back through the network during training by bypassing some layers which enables the network to learn more effectively and avoid getting stuck in suboptimal solutions.

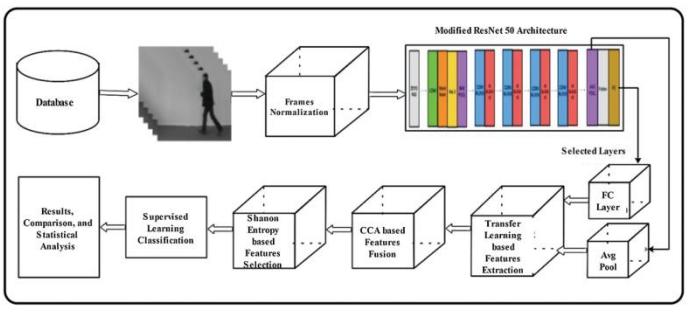

Fig 2: proposed deep learning-based architecture for human action recognition

4. **Enhanced Feature Fusion:** Four convolutional layers are applied to the concatenated feature maps. These layers further refine the representation and facilitate better (segmentation) information fusion, leading to improved detection and localization performance.

5. **Detection Head:**The highest-resolution information (1/32 of the original image) is fed into a Pyramid Pooling Module (PPM) head,

This module aggregates information from different spatial scales, allowing the network to capture global context and context-aware features.
A detection head is attached to the output of the PPM head. This head analyzes the pooled information and outputs a binary result: "real" or "fake" for the entire image.To be more specific , The Pyramid Pooling Module (PPM) organizes the process between the encoder and decoder in a detailed manner to influence every single pixel in an image, starting by the "Encoder" phase where we find a feature maps from diverse convolutional network stages collection information from many varying scales with each pixel mirroring details about its local context, through maximum pooling at varying scales, the spatial dimension of the feature maps is decreased which is a result of the condensment of the captured details, the latter pooled maps are then linked to become a single feature maps full of details where each single feature map offers a full representation that is a combination between local details and broader context at each pixel.

The "Decoder" then takes the output of the "Encoder" as input, where each pixel represents a fusion of information from different scales. The "Deconvolution" will progressively upsample the map and increase the spatial converage by enhancing the detail caputred by each pixel, this tasks depends on the decoder's output which can either be a "segmentation mask", where pixels become assigned class labels delineating different objects in the iage or "Bounding Boxes" that define object borders based on the enriched feature map's information. The encoder and decoder collaboratively refine pixel-level information, contributing to the overall effectiveness of the Pyramid Pooling Module.

6.**Localization Head:** The remaining feature levels are processed through a Feature Pyramid Network (FPN), similar to the original UPerNet architecture. This FPN further refines the features at different scales. At the highest resolution, a localization head is added. This head analyzes the refined features and outputs a mask highlighting the manipulated areas in the image. This mask essentially provides a visual representation of the detected forgery. For further details, the FPN leverages a pre-trained convolutional neural network, referred to as the backbone, for feature extraction from the input image. This backbone can be any standard image classification model such as ResNet . It enhances the extracted features by creating a feature pyramid . This synergistic approach enables the FPN to preserve fine-grained details from higher resolution layers and incorporate broader semantic context from lower resolution layers. The resultant feature pyramid comprises multiple maps at distinct scales, with each map capturing specific details and context relevant to its scale.

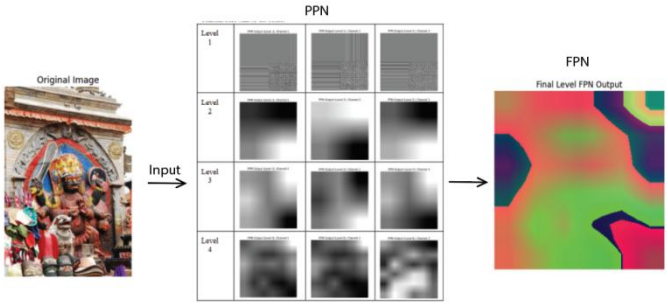

Fig 3 : PPM and FPN

7. **Loss function:** The final loss function of our network comprises two components: at the image level, we utilize binary cross-entropy (BCE) to ensure accurate binary detection results ($Lossclf$), while at the pixel level, Dice loss ($Loss\ seg$) is employed to measure the overlap between the generated mask and the ground truth, enhancing mask correctness. Unlike the original MVSS-Net, we omit edge loss, as it is unnecessary for our network. The overall loss (L) is calculated as a weighted sum: $\alpha \cdot Lossclf + \beta \cdot Lossseg$, with α and β set to 0.04 and 0.16, respectively, based on empirical studies to achieve an optimal trade-off between the two losses.

$$L = \alpha \cdot Loss_{clf} + \beta \cdot Loss_{seg}$$

III. Dataset Construction

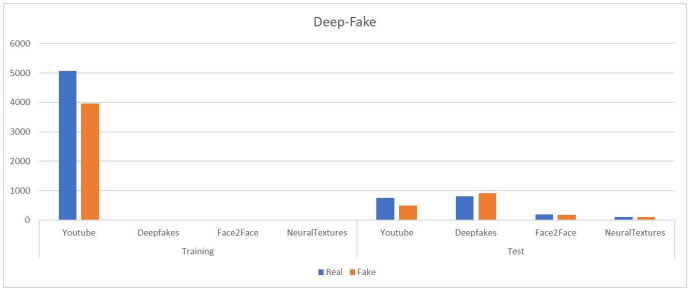

Fig4: DeepFake Dataset visualization

The researchers compiled a dataset for their study on detecting shallowfake and deepfake images. They used the CASIAv2 dataset for shallowfakes, which was divided into training, validation, and test sets. The deepfake dataset was constructed based on the FaceForensics++ dataset, with frames extracted from the videos. The dataset included a substantial number of authentic and forged frames, which were further divided into training, validation, and test sets. Ground-truth masks were generated for the deepfake dataset. However, during our experience with the datasets, we encountered a problem while attempting to download the NIST dataset. As a result, we reached out to the NIST Administration via email at mfc_poc@nist.gov for assistance. Additionally, the datasets had a large volume of 40GB, which posed challenges in running our code in platforms like Collab or locally. Therefore, we actively searched for alternative solutions to address this issue.

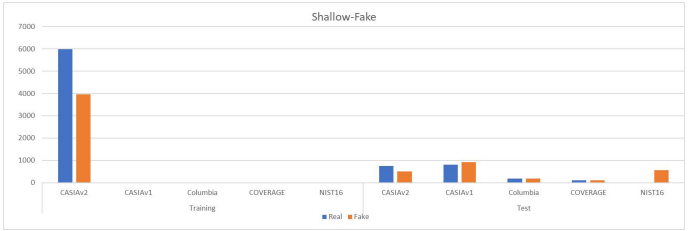

Fig5: Shallowfake Dataset visualization

III.Methodolgy:

After thoroughly reviewing the research paper, our initial step involved a comprehensive understanding of the provided code. We systematically dissected the code, isolating each function to gain insights into their specific roles. This process facilitated the comprehension of the workflow and enabled the identification of critical components. Afterwards, we employed visualization techniques to enhance our understanding of each function's outputs. This iterative step played a pivotal role in identifying potential issues and validating whether the functions behaved as anticipated.

The original research paper utilized powerful hardware, specifically four 32 GB NVIDIA V100 Volta GPUs . Therefore cannot be runin standard environments like Colab, adding to that the dataset's substantial size (40GB) and differing configurations.

Additionally, we encountered challenges related to the dataset itself. The lack of comprehensive documentation and details regarding the dataset's configuration posed significant hardships. This lack of information hindered our ability to seamlessly integrate the dataset into our work environment.

To address these challenges, we decided to allocate a virtual machine(8x Tesla v100 128Go) with significantly enhanced resources. This strategic move aimed to overcome the limitations posed by our previous computing environment and the dataset-related issues. The decision proved effective, accelerating the overall process. The training, which initially faced failure due to hardware and dataset constraints, was successfully completed on the allocated virtual machine, concluding in approximately two days.

Feature	32GB NVIDIA V100 VOLTA	8x Tesla v100 128Go
Total memory	32GB	1024GB
Streaming Multiprocessors	80 SMs	448 SMs
CUDA core	5,120 CUDA cores	28,672 CUDA cores
Tensor Core	640	640 Tensor Cores per GPU x 8 GPUs

Tab 1 : Comparisation table VM tesla vs NVDIA v100

Despite grappling with issues such as dataset configuration problems and insufficient documentation, our adapted approach enabled us to overcome these obstacles and produce results aligned with the research paper's objectives.

IV.Optimisation :

After successfully replicating the results of the previous research paper, we pursued various approaches to maximize the full potential of this work. Our experiment involved testing three different methods, which we are about to detail:

1/Architectural Modification: In this approach, we suggest a significant change to our model by replacing the Resnet50 backbone with an alternative structure better suited to our task. The goal is to identify a backbone architecture that improves our model's overall performance, maximizing its effectiveness in addressing the challenges of our problem.

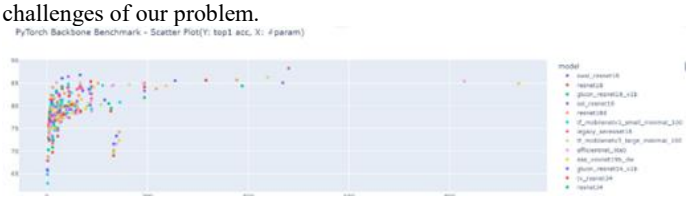

Fig 6 : PyTorch backbone ranked by parameters

complex models, while potentially offering better performance, require increased computational resources and training time.

The decision to explore alternative backbones needs careful consideration, balancing potential benefits against practical constraints. While not an immediate solution, this approach holds promise for future enhancements.

As depicted in the figure, various pre-trained backbones outperform Resnet50. However, it's crucial to recognize that while these alternatives may excel in specific scenarios, their effectiveness may not seamlessly translate to our unique use case. This difference could be due to the specific nature of our task, where Resnet50 may have advantages in certain characteristics or patterns. Additionally, more complex models, while potentially offering better performance, require increased computational resources and training time.

Despite the mixed results of substituting Resnet50 with different models, as highlighted in the figure, we must acknowledge that this strategy has proven effective in some cases. However, it comes with increased resource demands such as computational requirements and longer training times. Therefore, the decision to explore alternative backbones requires careful consideration, weighing potential benefits against practical constraints like computational resources and training duration. Although not an immediate solution, this avenue shows promise for future improvements.

2/Hyperparameter tuning : Our second approach involves retraining the model by adjusting existing parameters and introducing new ones. We kicked off by modifying the following key settings: We changed the image_size to 256 and adjusted Batch_size to 32 to speed up the training process. The parameters lambda_seg and lambda_clf were set to 0.1 each to strike a balance between classification loss and segmentation loss. Initially, we used "adam" as the optimization algorithm, and in a subsequent experiment, we introduced "RMSprop" due to its faster convergence and adaptive learning rate properties.

Recognizing that prior researchers likely explored various hyperparameter combinations for optimal results, we acknowledge the potential benefits of this method. However, we are mindful that it may be computationally expensive and could yield less favorable outcomes compared to the results presented in the original research paper. Therefore, we carefully weighed both the potential advantages and costs before making adjustments to the established hyperparameter settings.

3/DATASET:

In our final and most focused method, we noticed that our model works better on deepfake images compared to shallowfakes. So, we suggest tweaking the model to better catch shallowfakes by fine-tuning it on datasets specifically designed for shallowfake detection.

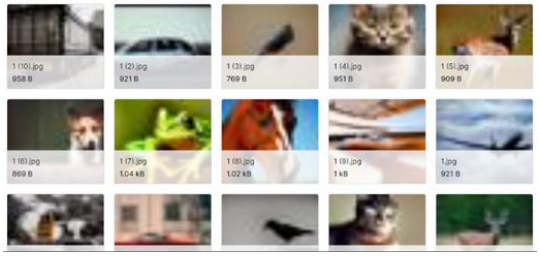

Fig 7 :CIFAKE Dataset

To start this process, we actively looked for a dataset that fits our needs. We found one called CIFAKE on Kaggle, containing two classes: real and fake. For the real images, we used data from Krizhevsky and Hinton's CIFAR-10 dataset. For the fake images, we generated a counterpart of CIFAR-10 using Stable Diffusion version 1.4. The dataset includes 100,000 training images (50,000 per class) and 20,000 testing images (10,000 per class). This fine-tuning strategy helps our model adapt and become more effective in spotting shallowfakes in various situations.

4/Data preparation

The next step is to prepare the data using tasks such as consolidating all images into a single folder and generating a paths.txt file .. to the perform finetuning

5/Training :

After generating the paths file we perfomed the training phase

6/Evaluation :

For the evaluation, we first checked the model using only the shallowfake dataset, and it showed an accuracy of 0.66, sensitivity (sen) of 0.60, specificity (spe) of 0.9565, F1 score of 0.8437, and an area under the curve (AUC) of 0.88.

\DataSets Metrics\	ShallowFakes	Both
Pixels	0.8044	0.8044
Inf	0.60	0.8451
f1	0.8437	0.861
AUC	0.88	0.9331

Tab2: Dataset Metric Values

Then, we expanded the evaluation to include both shallowfake and deepfake datasets. The results were as follows: accuracy of 0.8123, sensitivity of 0.8451, specificity of 0.9565, F1 score of 0.861, and AUC of 0.9331. With a threshold of 0.5000, the combined F1 score reached 0.8236.

Comparing the two sets of results, there's a significant improvement in metrics, especially in accuracy and the area under the curve.

In summary, our meticulous fine-tuning strategy, leveraging a pre-trained model and extensive data, has resulted in outstanding performance, particularly excelling in AUC and F1 score metrics, showcasing the robustness and versatility of our approach across diverse datasets.

CONCLUSION :

The success of the Shallowfakes data upgrade in improving model performance is evident, and the incorporation of "Hyperparameter Tuning" and "Modifying the architecture" using alternative models further enhances our system. Looking ahead, combining these optimization methods holds promise for even better results, potentially refining the Area under the Curve (AUC) and overall model performance. This integrated approach reflects our commitment to pushing the boundaries of achievable results in Deep Learning, showcasing dedication to excellence and innovation in optimization strategies.

References:

Fig1: https://blog.csdn.net/weixin_43780665/article/details/134177973 Fig2:https://www.researchgate.net/figure/Proposed-deep-learning-based-architecture-for-human-action-recognition_fig1_354091072
UPerNet: https://arxiv.org/abs/1511.06440
ResNet50: https://arxiv.org/abs/1512.03385
Pyramid Pooling Module (PPM): https://arxiv.org/abs/1807.06521
Feature Pyramid Network (FPN): https://arxiv.org/abs/1612.03144