Learning Document Graphs with Attention for Image Manipulation Detection

Hailey James, Otkrist Gupta, and Dan Raviv

Lendbuzz, Boston, MA 02110, USA
Correspondence: `otkrist.gupta@lendbuzz.com`

Abstract. Detecting manipulations in images is becoming increasingly important for combating misinformation and forgery. While recent advances in computer vision have lead to improved methods for detecting spliced images, most state-of-the-art methods fail when applied to images containing mostly text, such as images of documents. We propose a deep-learning method for detecting manipulations in images of documents which leverages the unique structured nature of these images in comparison with those of natural scenes. Specifically, we re-frame the classic image splice detection problem as a node classification problem, in which Optical Character Recognition (OCR) bounding boxes form nodes and edges are added according to a text-specific distance heuristic. We propose a Variational Autoencoder (VAE)-based embedding algorithm, which when combined with a graph neural network with attention, outperforms both a state-of-the-art image splice detection method and a document-specific method.

Keywords: Manipulation Detection · Graph Neural Networks · Variational Auto-Encoders.

1 Introduction

Identifying spliced images, or images in which content has been added during post-processing, has become an important area of research in computer vision. While spliced images of natural scenes can be used to propagate fake news, many real-world image splicing examples occur in images containing primarily text, or images of documents. Digital documents are commonly used to verify information in domains such as finance and administration, which creates a natural opportunity for forgery and manipulation. With a background of largely white space, images of digital documents pose a problem for most state-of-the-art image manipulation detection models, as they are traditionally trained to detect inconsistencies in texture and background features. In addition, the forged region of a document often looks very similar to the original image, as both often are simply black text on a white background.

In this paper, we propose an image manipulation detection system designed for improved performance on digital documents. Unlike in natural scene images, forgery detection in digital documents allows us to use Optical Character Recognition (OCR) to narrow the search space of potential manipulations to the textual content of the image. As the number of text boxes on a given page varies and the existence of meaningful relationships between boxes can be inferred, the resulting OCR bounding boxes are natural candidates for learning with Graph Neural Networks (GNNs). Specifically, we can construct a graph G from a single image of a document in which the OCR bounding boxes form nodes $\mathbf{v} = \{v_1, v_2, v_3, ...v_{|V|}\}$ where $|V|$ is the number of detected OCR text boxes on the page. The nodes are connected by edges using a document-specific distance heuristic, such that

$$e_{v_i,v_j} = \begin{cases} 1, & \text{if } d_{v_i,v_j} \leq T \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where d_{v_i,v_j} is the distance between nodes v_i and v_j, T is a numeric threshold tuned such that connected edges exist within the same paragraph or textual region on a page. We can then re-frame the splice detection in digital documents as a node classification problem, in which text nodes are classified either spliced or genuine.

The primary challenge for this approach is initializing the node embeddings to effectively create feature vectors $\mathbf{h} = \{\vec{h}_1, \vec{h}_2...\vec{h}_{|V|}\}$ for effective splice manipulation detection. Most methods for learning text-based features from images (such as OCR) are designed to extract high-level features such as character values while ignoring low-level features such as text blur, spacing, or font properties. However, these are the exact properties which are most likely to be relevant for identifying a spliced box, while high-level features such as the character value or word are less likely to be discriminative. In the case of manipulation detection we are interested specifically in ignoring high-level features such as text content while extracting low-level character-based features.

To accomplish this task, we designed a feature extraction algorithm based on pre-training the embeddings using a Variational Autoencoder (VAE) on a set of input and output image pairs. In each image pair, the content varies while the low-level features (character weight, font, spacing, etc) are held constant. This results in a latent space embedding that is sensitive to manipulation artifacts while agnostic to the text content. The encoder from this feature extraction model forms the first part of the manipulation detection model, with weights pre-trained on the specially-designed images pairs.

An additional challenge is to design a mechanism which leverages the context of a given block of text for improved classification. In the case of forgery detection, the relationships between text nodes are particularly important, as an anomalous node is most likely to be detected by comparing its features with those of its neighbors. We would expect to see subtle feature differences in spliced text, as it likely would have been spliced from another document or image with different character-level features.

Our solution involves adapting a graph attention layer [31], in which attention weights are trained to update the node embeddings according to the edges between nodes $\mathbf{h} = \{\vec{h}_1, \vec{h}_2...\vec{h}_{|V|}\} \rightarrow \mathbf{h}' = \{\vec{h'}_1, \vec{h'}_2...\vec{h'}_{|V|}\}$. This allows our model to appropriately update a node embedding by attending to its neighbors for binary classification (see Figure 1).

We evaluate our method on a public datasets of programmatically-generated document splice dataset. Our proposed model outperforms both a state-of-the-art image splice detection method and a document-specific method, both of which are generally unable to differentiate the spliced text from the genuine text.

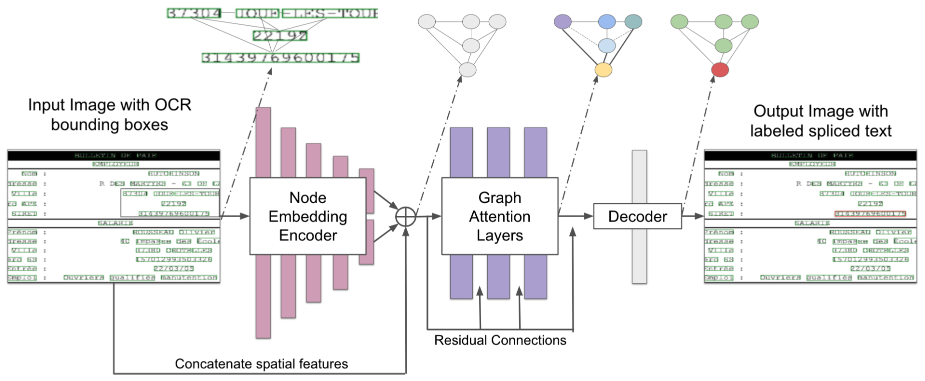

Input Image with OCR bounding boxes
Node Embedding Encoder
Graph Attention Layers
Decoder
Output Image with labeled spliced text
Residual Connections
Concatenate spatial features

Fig. 1. Architecture overview. Our proposed system is designed to label text on an image of a document as original to the document or as spliced (copied and pasted from another image). Left to right: First Optical Character Recognition (OCR) bounding boxes are obtained, and a graph structure is formed by treating the text boxes as nodes and adding edges accordingly to a document-based distance heuristic. Each graph is passed through the pre-trained encoder, embedding each node based on low-level features (Section 2). The graph's spatial features are then concatenated with the encoder output, as represented by $\bigoplus$. Next, the node embeddings are updated by a trained graph attention layer, which implicitly weights edges between nodes to encode relational information. Finally, a decoder classifies each node as spliced or genuine, at which point they can be mapped back onto the input image.

2 Related Work

Image Splice Detection Image splicing is a manipulation operation in which content is copied and pasted from a source image onto a destination image. Current methods for detecting splices in images can be broadly categorized as methods that search for local inconsistencies in the image background noise [22,26] or compression discrepancies [16,12,23], methods that leverage camera-based artifact inconsistencies [17,27,32], camera-based methods [9] or deep-learning based methods [2,33,18,9,11], with deep-learning based methods generally achieving the best performance. Several public datasets for benchmarking performance on image splice detection in natural scenes have been released [20]. In this paper, we compare our model with Self Consistency [18], a state-of-the-art image splice detection method with a publicly available trained model.

Document Manipulation Detection Methods for detecting manipulations in documents often try to leverage document-specific features. Examples include methods that use intrinsic document elements, or portions of the document that can be matched against a template to gauge authenticity [1], font-based features [4,10,8], alignment or skew-based features [6] For images of physical documents, this has included printer identification [5,29,28,13]. [30] present findings from deploying manipulation techniques on real-world datasets.

Graph Neural Networks Graph Neural Networks (GNNs) are intended to deal with arbitrarily structured data in order to generalize learning methods to the non-Euclidean domain [7]. Typically, Graph Neural Networks perform an iterative process of updating node states until equilibrium is reached. The resulting node states can be learned by consecutive trainable layers in order to classify nodes, graphs, or edges. [31] proposed a Graph Attention Networks (GATs) for learning an attention mechanism on the relationships between nodes.

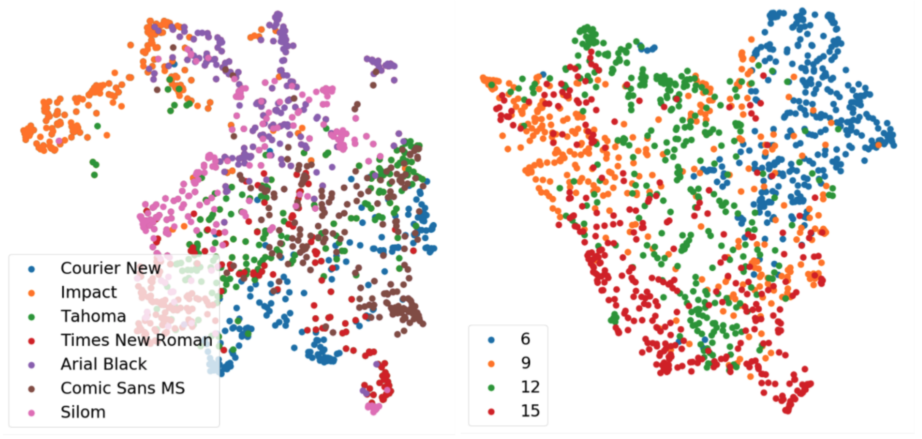


Fig. 2. Encoder embeddings capture low-level font features. UMAP [24] visualization of randomly-generated text node embeddings labeled by font (left) and font size (right). The VAE is able to separate text images based on these low level features. Notably, the embeddings reveal groups based on character size, despite each text box being scaled to a uniform height and cropped to a uniform width before being passed to the model.

3 Methods

Node Embedding Initialization One of the primary challenges in learning the document graph for forgery localization is initializing the node embeddings.

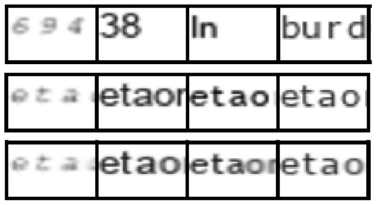


Fig. 3. Data used for pre-training the encoder. VAE inputs (top), VAE outputs (middle), and VAE reconstructions (bottom). The inputs are constructed by drawing random strings of text (not OCR text images) from OCR results on the training images. The input images are constructed by assigning the random string specific properties, and the outputs are always the same string ("etaonisrhldcmufpgwybvkjxqz") with matching properties to the input. This pairing enables the encoder to capture low-level manipulation artifacts while ignoring content-like character value or sequence patterns.

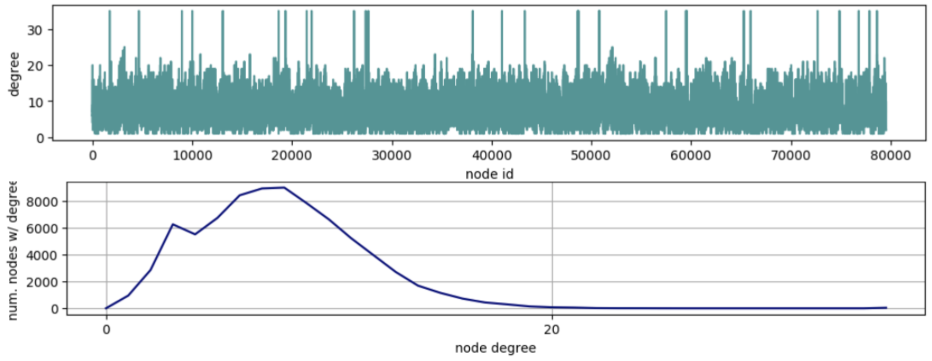


Fig. 4. Graph statistics on a sample of 256 images from the Auto-Splice Dataset. Top: Degree by number of nodes with that degree on a sample set. Bottom: Number of nodes by degree. The median degree is 8, which intuitively makes sense as the number of nearby words that might be examined to determine if a block of text has been spliced.

The node embeddings must have uniform size and contain information that is likely to be useful for manipulation detection. While we can naïvely obtain OCR bounding boxes to represent each node, bounded regions of pixel values will be variable in size and obscure important information features. When text is spliced from one image to another, it is unlikely that features like the character size, font, spacing, and aspect ratio will be identical from the source to the destination. Thus while most learning methods using images of text seek to extract the high-level features such as character value or words (i.e. using OCR), we are interested in an embedding algorithm which can capture these low-level features. One option would be to try to directly learn the fonts, character spacing values, and font sizes and form a corresponding feature vector. However, this would require constructing a pre-defined set of fonts and other low-level features and would be unlikely to generalize well between document sets.
Instead, we train a generative model on image pairs in which the content (character values) change but low-level forgery related features are held constant (see Figures 3 and 2. Ablation studies show improvements of a Variational Autoencoder (VAE) over a simple auto-encoder and no pretraining, indicating that a VAE architecture constrains our model to form smooth latent space representations that are conducive to interpolation when unseen data is encountered. In each image pair, the input is formed by sampling a random string from the text in the training images (as obtained through OCR) and creating an image formed from this text and a random character spacing, size, font, and level of Gaussian blur. The output image is formed by taking a fixed string (the same string for every output) and creating an image with the same character spacing, size, font, and blur values as the input. We then take a random crop of 30x50 pixels of the input image and a fixed crop (from the top left corner) of the output image of the same dimensions. While any fixed string could be used as the output string, we chose to use the alphabetical characters ordered by frequency of use ("etaonisrhldcmufpgwybvkjxqz") in the English language. Compared with a traditional non-variational autoencoder, we found that the model performed better on "out of distribution" data, or when the manipulated dataset included fonts or variations not found in the pre-training data.
Our VAE architecture is composed of four fully-connected encoding layers (sizes 1500, 1204, 908, and 612) with ReLU activation and four fully-connected decoding layers (size 612, 908, 1204, and 1500) with ReLU activation, with a latent dimension of size twenty. We use a final Sigmoid activation function, Xavier initialization, batch size 32, learning rate 0.0001. The loss function is composed of the reconstruction loss and a regularization term, the Kullback-Leibler (KL) divergence between the means and variances encoded in latent space and that of a unit Gaussian. To train our model, we adapt the approximation proposed in [19]

$$\sum_j KL(q_j(z|x) \parallel \mathcal{N}(0,1)) + \mathbf{L}(x,\hat{x})$$

where $\mathbf{L}$ is the reconstruction error, j is the latent space dimension, KL is the KL divergence, $q_j(z|x)$ is the encoded distribution, and $\mathcal{N}(0,1)$ is a unit Gaussian. To train our model, we adapt the approximation proposed in [19], or

$$\mathcal{L}(\theta; \hat{x}^{(i)}, x^{(i)}) \simeq \frac{1}{2}\sum_{j}(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2)) + BCE_\theta(\hat{x}^{(i)}, x^{(i)})$$

where $\mathcal{L}$ is the loss given the parameters θ on a text image $x^{(i)}$ from the synthetic dataset, $\mu_j^{(i)}$ and $\sigma_j^{(i)}$ are the mean and variance of the distribution j on image i in the latent space, and $BCE_\theta(\hat{x}^{(i)}, x^{(i)})$ is the binary cross entropy loss between the output text image $x^{(i)}$ from image pair i and the reconstructed image $\hat{x}^{(i)}$.

Graph Learning and Classification We construct the graph by adding edges between OCR bounding boxes given a document-based heuristic (see Equation 1). Because boxes can vary significantly in size, considering a midpoint to midpoint euclidean distance d is not practical for adding edges, as larger boxes would be biased towards having a lower degree than smaller boxes. Instead, we propose an edge addition formula which adds edges based on a function of vertical and horizontal gaps between the bounding boxes and the respective widths and heights of each box. Specifically, edges are added when the vertical gap g_v between the boxes is less than three times the minimum height h_{min} and the horizontal gap g_h is less than four times the minimum height h_{min} (see Equation 2). Additionally, the degree of all nodes is capped at 35. See Figure 4.

$$e_{v_i,v_j} = \begin{cases} 1, & \text{if } g_v \leq 3 \times h_{min} \cap g_h \leq 4 \times h_{min} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The nature of text on a page means that, for the purpose of splice detection, not all edges can be considered equal. For example, a block of text that differs from the rest of the text on a horizontal line may be more likely to have been spliced than a block of text that differs from the text above or below. In order to learn these relationships and implicitly apply different weights to different edges, we developed a model based on graph attention layers, as proposed in [31]. The graph attention layer takes as input the set of node features , $\mathbf{h} = \{\vec{h}_1, \vec{h}_2 ... \vec{h}_{|V|}\}, \vec{h}_i \in \mathbb{R}^{F_n}$, where $|V|$ is the number of nodes in the graph and F_n is the node embedding dimension at the nth graph attention layer. In the first graph attention layer, the input is the encoder latent space embedding concatenated with the scaled bounding box coordinates of the text nodes, or $2 \times \textit{latent dim} + 4$. The outputs of a graph attention layer are an updated set of node features, potentially with a new node embedding dimension, or $\mathbf{h}' = \{\vec{h'}_1, \vec{h'}_2 ... \vec{h'}_N\}, \vec{h'}_i \in \mathbb{R}^{F_{n+1}}$.

An attention weight matrix, $\mathbf{W} \in \mathbb{R}^{F_n \times F_{n+1}}$, performs self-attention on each of the nodes by sharing the attention mechanism a for coefficient computation. While the weight matrix is applied over all nodes in the graph, structural information is preserved by masking the attention to include only first-order neighbors of a node i. Attention is normalized across all edges by using the softmax function.

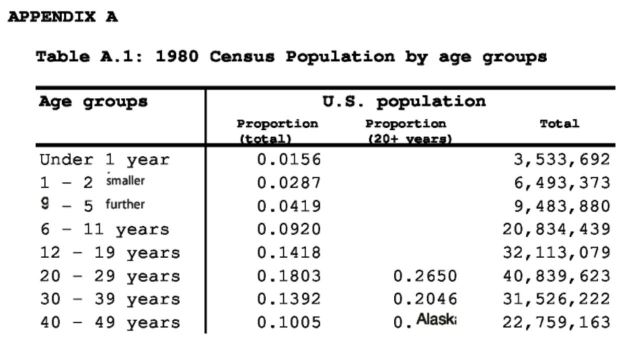
In our proposed model, the attention mechanism is a single-layer neural network with Leaky ReLU activation.

$$\alpha_{ij} = \frac{\exp(a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j))}{\sum_{z \in N_x} \exp(a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_k))}$$

The normalized attention coefficients serve to form a linear combination of input features, and form the final output node embeddings for the layer. We found that using multi-head attention and ELU activation improved training stability, consistent with results in [31]. The output features can thus be represented by

$$\vec{h'}_i = \Vert_{k=1}^{K} ELU(\sum_{j \in N_i} \alpha_{ij}^{k} \mathbf{W}^k \vec{h}_j)$$

where K represents the independent attention heads, and $\Vert$ denotes the concatenation of the K outputs, or averaging in the case of the final graph attention layer.
The model architecture is composed of four fully-connected encoding layers with 1500, 1204, 908 and 612 nodes respectively (Section 2). The encoder is followed by a three graph attention layers of sizes 36, 29 and 22 and with 4,4, and 6 attention heads respectively. Residual connection are added between graph attention layer. Finally the updated embeddings are classified by a decoder with a single fully connected layer of size 22. We use simple binary cross-entropy loss for training the model end to end. See Figure 1 for a visualization of the model pipeline.
Auto-Splice Dataset While many publicly available datasets exists for bench-



APPENDIX A

Table A.1: 1980 Census Population by age groups

Age groups	U.S. population		
	Proportion (total)	Proportion (20+ years)	Total
Under 1 year	0.0156		3,533,692
1 - 2 smaller	0.0287		6,493,373
9 - 5 further	0.0419		9,483,880
6 - 11 years	0.0920		20,834,439
12 - 19 years	0.1418		32,113,079
20 - 29 years	0.1803	0.2650	40,839,623
30 - 39 years	0.1392	0.2046	31,526,222
40 - 49 years	0.1005	0. Alask	22,759,163

Fig. 5. An example image from the spliced image dataset with three splices ("smaller", "further" and "Alaska." In order to classify each block of text as spliced or genuine, the model is trained to recognize the subtle variations in font or character width in comparison to its surrounding text.

marking image splice detection, there are few datasets designed for investigating manipulations in images of text-based documents. As a result, we have created

and release for public use a dataset of spliced text in images of non-private documents. The original source of the documents is PDF files from the ICDAR 2013 Table Competition [14].[1] The original PDFs contain a wide variety of text types, ranging from paragraphs in articles to numeric tables. This property helps ensure a wide variety of character-level features both within and between documents. Although care was taken to design an automatic splicing process for realistic splices, it is possible that the automation process introduces biases compared with a dataset of hand-spliced text boxes. To account for this, we manually inspected the dataset and found that it contains a range of difficulty levels as judged by human detection ability. Figure 5 displays a "typical" easy/medium difficulty image based on visually inspecting a random subset of samples.
First, PDF-level word boxes are extracted from the PDF document directly. Importantly, the boxes are extracted directly from the PDFs and *not* using OCR, which would otherwise give our model an advantage by being able to bound each potential spliced region with certainty. Each page is then converted to a PNG image to form a set of unmanipulated document images. To calculate the number of splices that will occur in the dataset, we multiply the number of PDF word boxes in the dataset by 0.05, such that in the dataset overall, 5% of the PDF boxes will be spliced. To perform each splice, a source image is chosen at random, from which a source box is drawn. A destination box and destination image are similarly drawn at random, and the source box is scaled and cropped in order to fit the dimensions of the destination box and pasted. This process is repeated until the specified number of splices have been performed, with the constraint that every document retains at least 85% of its original text boxes. Splicing does not occur between instances of the same document, and there is no constraint that enforces boxes to be spliced from a document with different character-level features (font, font size, etc). From each of the 200 total PDF pages, we create 50 identical images to serve as an initial images. The required number of splicing operations is performed based on the percentage of splices – we created datasets of both 1% and 5% spliced text boxes, as is reported in Table 1. The resulting dataset has 10,000 unique images of size 1700x2200, complete with ground truth splice bounding boxes.

4 Experiments

We experimented with a variety of architectures and ranges for the hyperparameters, including an encoder with up to 16 fully-connected layers, up to 5 graph attention layers, up to 4 fully-connected decoding layers, and the presence or absence of residual connections, and ablation studies for the VAE-based pretraining. We note that our architecture exploration was far from exhaustive and that further experimentation in architecture design could lead to improved performance.

[1] The images in this dataset are images from PDFs from academic works. The PDFs include articles from sociology journals, some of which discuss violent content and may be upsetting to certain readers.

Table 1. The results of our model compared with two other splice detection models on our dataset of digital document splices. The state-of-the-art image manipulation detection models are in effect unable to identify splices in documents, despite being far more computationally expensive than our proposed model. Each model was run on a single NVIDIA GeForce 29C RTX P8 GPU, which is reflected in the mean seconds per image.

	Auto-Splice (5% Manip)	Auto-Splice (1% Manip.)	Auto-Splice
	Test F1		Mean Seconds per Image
Self Cons.	0.171	0.130	104.40 s
Intrin. Doc.	0.559	0.348	17.64 s
Proposed Model	**0.904**	**0.653**	**8.01 s**

Comparison Models We compare our proposed model with two state-of-the-art splice detection models, Intrin. Feats [3] and Self Consistency [18]. [3] is a model especially designed for detecting manipulations in images of documents. Similar to our proposed model, it starts from OCR bounding boxes obtained through Tesseract OCR. Following OCR extraction, a feature vector for each text bounding box is constructed using the character size, principal inertia axis, horizontal alignment, and Hu moments [15]. One of the main contributions of the method is the use of character-level features that are intrinsic to the document, in comparison with other methods which rely on the presence of extrinsic feature watermarks or a master template. [18] is a deep learning based method designed to search for inconsistencies in the low-level pixel information for the purpose of splice detection. It has achieved state-of-the-art performance on several challenging splice datasets of images of natural scenes, such as *Columbia* [25] and *Realistic Tampering* [21] . We made use of the publicly available code and pre-trained model for evaluation released by the authors to perform the comparison on the spliced document image datasets.

Results While object detection methods often report a mean Average Precision (mAP) score, this would primarily measure the quality of the OCR engine rather than splice detection model performance. This is because the inputs to our model and [3] are OCR bounding boxes. Instead, to evaluate our proposed model's ability to detect splices in digital documents, we measure both the F1 score with respect to labeling boxes as manipulated or genuine. Because our model takes inputs already partitioned by OCR bounding boxes, it is straightforward to calculate the F1 score given the predicted labels and ground truth labels.

Discussion We find our model compares favorably to both comparison models, both in Test F1 score and in Mean Seconds per Image. Future work could consider new graph construction methods, including methods which are trainable alongside the rest of the network. One example could include adding edges between blocks of text that are the same throughout the document. Additionally, we anticipate significant improvement could come from employing a more robust OCR method.

References

1. Ahmed, A.G.H., Shafait, F.: Forgery detection based on intrinsic document contents. In: 2014 11th IAPR International Workshop on Document Analysis Systems. pp. 252–256. IEEE (2014)
2. Bayar, B., Stamm, M.C.: A deep learning approach to universal image manipulation detection using a new convolutional layer. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security. pp. 5–10 (2016)
3. Bertrand, R., Gomez-Kramer, P., Terrades, O.R., Franco, P., Ogier, J.M.: A system based on intrinsic features for fraudulent document detection. In: 2013 12th International conference on document analysis and recognition. pp. 106–110. IEEE (2013)
4. Bertrand, R., Terrades, O.R., Gomez-Kramer, P., Franco, P., Ogier, J.M.: A conditional random field model for font forgery detection. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR). pp. 576–580. IEEE (2015)
5. van Beusekom, J., Shafait, F., Breuel, T.M.: Automatic authentication of color laser print-outs using machine identification codes. Pattern Analysis and Applications **16**(4), 663–678 (2013)
6. van Beusekom, J., Shafait, F., Breuel, T.M.: Text-line examination for document forgery detection. International Journal on Document Analysis and Recognition (IJDAR) **16**(2), 189–207 (Jun 2013). https://doi.org/10.1007/s10032-011-0181-5, https://doi.org/10.1007/s10032-011-0181-5
7. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
8. Chernyshova, Y.S., Aliev, M.A., Gushchanskaia, E.S., Sheshkus, A.V.: Optical font recognition in smartphone-captured images and its applicability for id forgery detection. In: Eleventh International Conference on Machine Vision (ICMV 2018). vol. 11041, p. 110411J. International Society for Optics and Photonics (2019)
9. Cozzolino, D., Verdoliva, L.: Noiseprint: A cnn-based camera model fingerprint. IEEE Transactions on Information Forensics and Security **15**, 144–159 (2019)
10. Cruz, F., Sidere, N., Coustaty, M., D'Andecy, V.P., Ogier, J.M.: Local binary patterns for document forgery detection. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 1223–1228. IEEE (2017)
11. Ghosh, A., Zhong, Z., Boult, T.E., Singh, M.: Spliceradar: A learned method for blind image forensics. In: CVPR Workshops. pp. 72–79 (2019)
12. Gupta, A., Saxena, N., Vasistha, S.: Detecting copy move forgery using dct. International Journal of Scientific and Research Publications **3**(5), 1 (2013)
13. Gupta, S., Kumar, M.: Forensic document examination system using boosting and bagging methodologies. Soft Computing **24**(7), 5409–5426 (Apr 2020). https://doi.org/10.1007/s00500-019-04297-5, https://doi.org/10.1007/s00500-019-04297-5
14. Hassan, T.: Icdar 2013 table competition dataset (2013), https://www.tamirhassan.com/html/competition.html
15. Hu, M.K.: Visual pattern recognition by moment invariants. IRE transactions on information theory **8**(2), 179–187 (1962)
16. Hu, W.C., Chen, W.H.: Effective forgery detection using dct+ svd-based watermarking for region of interest in key frames of vision-based surveillance. International Journal of Computational Science and Engineering **8**(4), 297–305 (2013)

17. Hu, W.C., Chen, W.H., Huang, D.Y., Yang, C.Y.: Effective image forgery detection of tampered foreground or background image based on image watermarking and alpha mattes. Multimedia Tools and Applications **75**(6), 3495–3516 (2016)
18. Huh, M., Liu, A., Owens, A., Efros, A.A.: Fighting fake news: Image splice detection via learned self-consistency. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 101–117 (2018)
19. Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014)
20. Kniaz, V.V., Knyaz, V., Remondino, F.: The point where reality meets fantasy: Mixed adversarial generators for image splice detection (2019)
21. Korus, P., Huang, J.: Evaluation of random field models in multi-modal unsupervised tampering localization. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–6 (2016). https://doi.org/10.1109/WIFS.2016.7823898
22. Liu, B., Pun, C.M.: Splicing forgery exposure in digital image by detecting noise discrepancies. International Journal of Computer and Communication Engineering **4**(1), 33 (2015)
23. Mayer, O., Stamm, M.C.: Exposing fake images with forensic similarity graphs. IEEE Journal of Selected Topics in Signal Processing **14**(5), 1049–1064 (2020)
24. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2018), `http://arxiv.org/abs/1802.03426`, cite arxiv:1802.03426Comment: Reference implementation available at http://github.com/lmcinnes/umap
25. Ng, T.T., Chang, S.F., Sun, Q.: A data set of authentic and spliced image blocks. Columbia University, ADVENT Technical Report pp. 203–2004 (2004)
26. Pun, C.M., Liu, B., Yuan, X.C.: Multi-scale noise estimation for image splicing forgery detection. Journal of visual communication and image representation **38**, 195–206 (2016)
27. Roy, A., Dixit, R., Naskar, R., Chakraborty, R.S.: Copy-move forgery detection with similar but genuine objects. In: Digital Image Forensics, pp. 65–77. Springer (2020)
28. Shang, S., Kong, X., You, X.: Document forgery detection using distortion mutation of geometric parameters in characters. J. Electronic Imaging (2015). https://doi.org/10.1117/1.JEI.24.2.023008
29. Shang, S., Memon, N., Kong, X.: Detecting documents forged by printing and copying. EURASIP Journal on Advances in Signal Processing **2014**(1), 140 (2014)
30. Van Beusekom, J., Stahl, A., Shafait, F.: Lessons learned from automatic forgery detection in over 100,000 invoices. In: Computational Forensics, pp. 130–142. Springer (2012)
31. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
32. Wen, L., Qi, H., Lyu, S.: Contrast enhancement estimation for digital image forensics. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **14**(2), 1–21 (2018)
33. Wu, Y., AbdAlmageed, W., Natarajan, P.: Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9543–9552 (2019)