Document Tampering Detection, Honors Project Monsoon-2019

Shubh Maheshwari & Devansh Gautam

[a] *Project Website*
Demo Website

1. Introduction

Previous semester we worked on document tampering classification. We looked into ICLR-2018 findit challenge [1] and evaluated various methods used in the challenge.

This semester we looked into document tampering detection. Goal is find the region where the document is forged. The problem becomes challenging due to the lack of large datasets to train.

2. Method

Similar to Yashas' work [2] we want to overcome the barrier of insufficient data by creating large amount of synthetic images and using domain adaptation improve the precision of our CNN model.

2.1. Dataset

For detection we are using the Find-it dataset [1] which provides 100 images for training and 80 images for testing. Validation set is created by dividing the training set into 2 sets using **90-10** ratio.

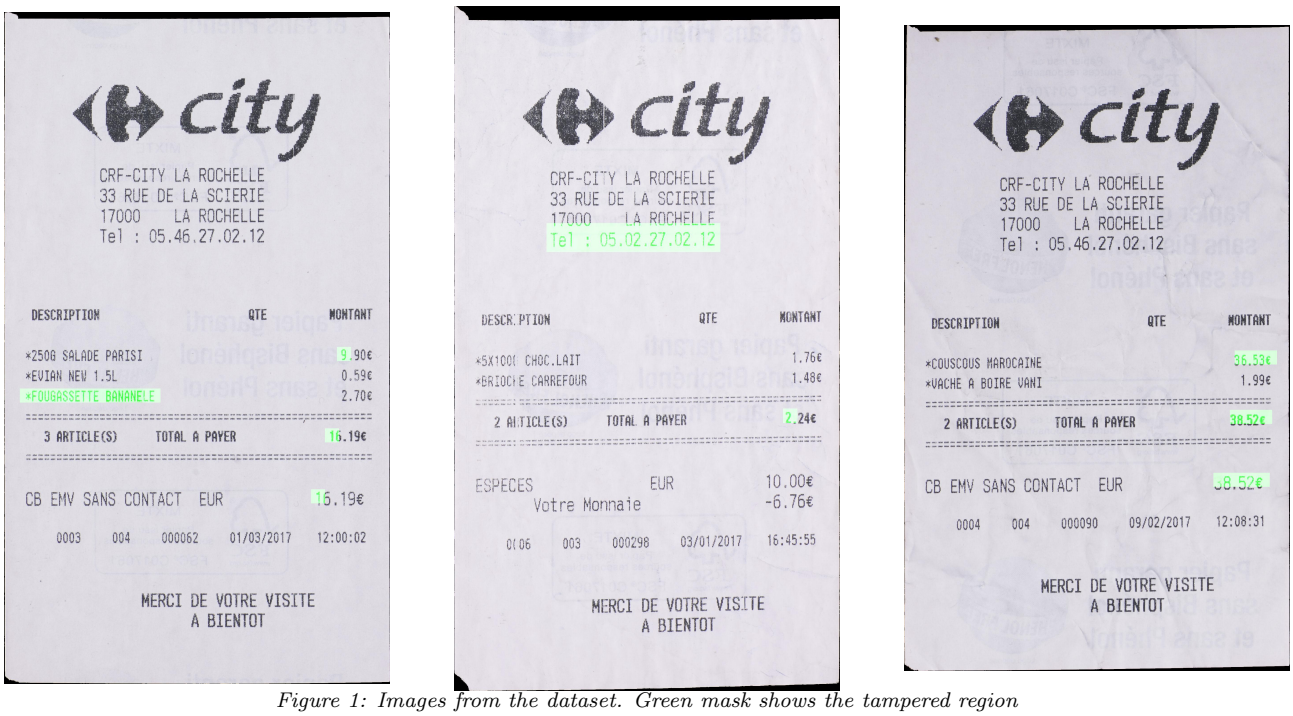

Figure 1: Images from the dataset. Green mask shows the tampered region

2.2. Synthetic Data creation

As tampered images are so low. It makes it very hard to train a CNN architecture with such data. Hence we create synthetic tampered images using 3 types of tampering.

1. In-painting
2. Copy-paste
3. Splicing

Find-it dataset provides 470 pristine images for classification. To create tampered images we first use a text detector [3] to extract text from the documents. To extract characters from the document, we use connected components. Finally different types of tampering are created by replacing the text region.

Thus, we create total of 5000 synthetic images. See figure-2 for examples of tampering.

Figure 2: Examples of synthetic data generated. (a)Red region was in-painted. (b) Green region was copied to red region. (c) Red region was copied from some another image

2.3. Creating Patches

The input to the model are 64x64 patches. To extract patches from the image. First, we run a text detector on the documents to detect the text. We use well known text detector by [3]. Then extract 64x64 patches are created by extracting the region around the text. This becomes the input to our model.

2.4. Class Imbalance

As a documents generally contains large amount of text out of which only a few regions are tampered. The data becomes highly imbalanced. In the find-it dataset we noticed that on average only 3.0% of patches extracted from an image were tampered. Hence during training tampered patches are **over-sampled**. We kept the ratio at ratio of 32:1. The chance of picking a tampered patch is 32 times more likely than a pristine patch.

2.4.1. Architecture

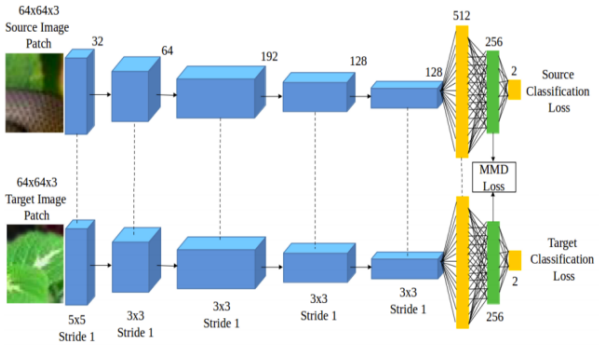

Figure 3: Model Architecture

As seen in the figure during training source model and target model are trained simultaneously. We are using an 5 layer CNN as the trunk of our architecture. The weights between them are shared. After passing it through 2 fully connected layers, to get a 256-dim representation for source and target images, between which the distance is minimized using MMD loss defined as:

$$L_m = min||\frac{1}{|X_s|}\sum_{x_s \in X_s}\Phi(x_s) - \frac{1}{|X_t|}\sum_{x_t \in X_t}\Phi(x_t)||^2 \quad (1)$$

where x_s and x_t are features for the source and target images respectively. $\Phi(x_s) and \Phi(x_t)$ are the calculated by passing the representation through Gaussian kernel.

3. Experiment

3.1. Training

Figure 4 shows the training and validation loss with and without using augmented data.

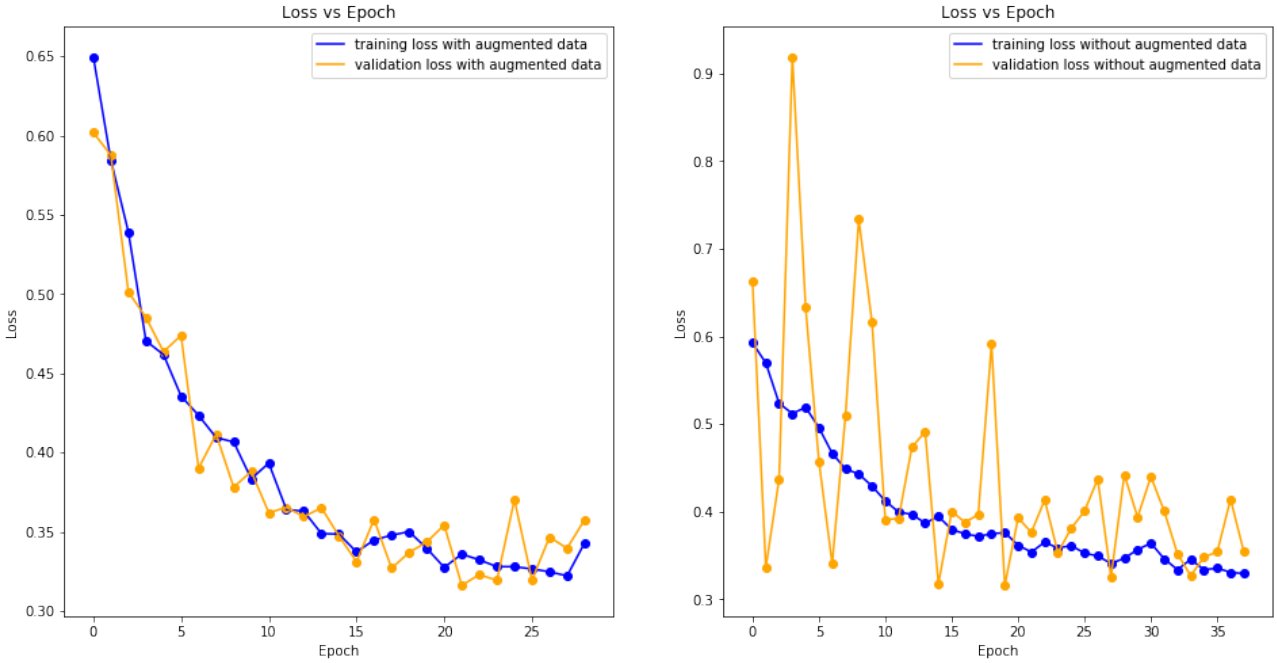

Figure 4: Train & Validation Loss

Figure 5 shows the training and validation accuracy with and without using augmented data. Here validation accuracy on the find-it images is shown by pink color.

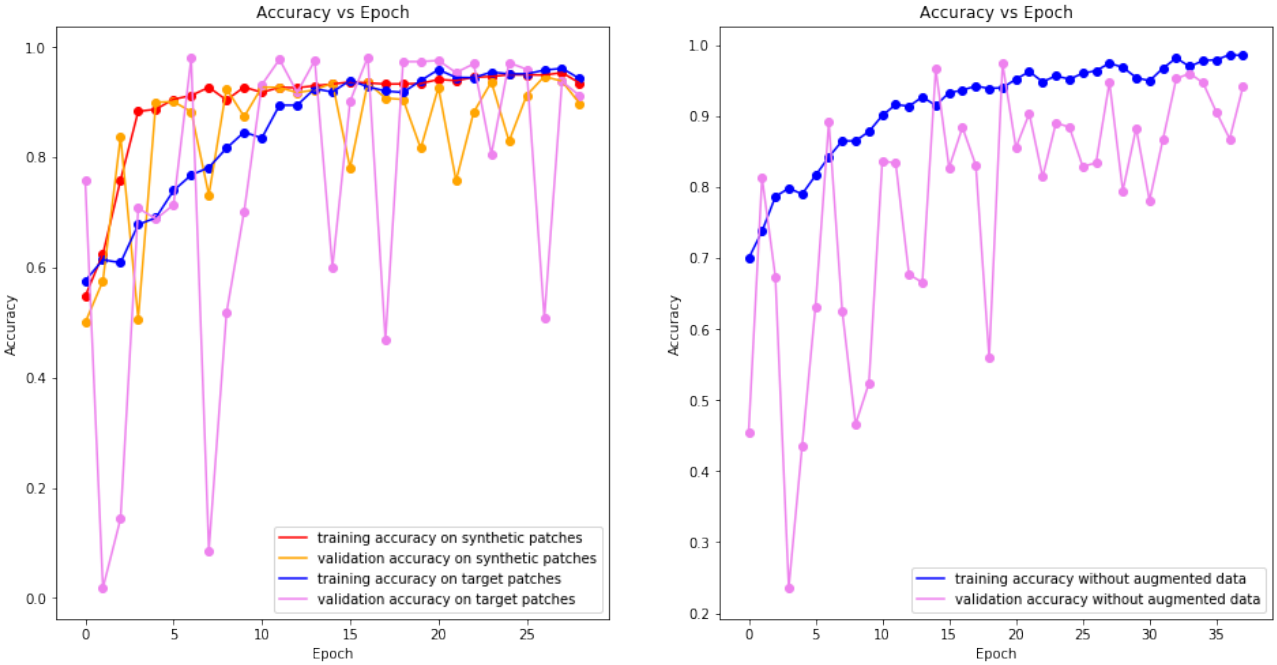

Figure 5: Train & Validation Accuracy

As accuracy is not a good metric for evaluating hence we are also measuring the f1 score during training. Figure 6 shows the training and validation f1-scores with and without using augmented data. It can clearly be seen that the model has over-fitted on the training data and hence is performing poorly on validation.

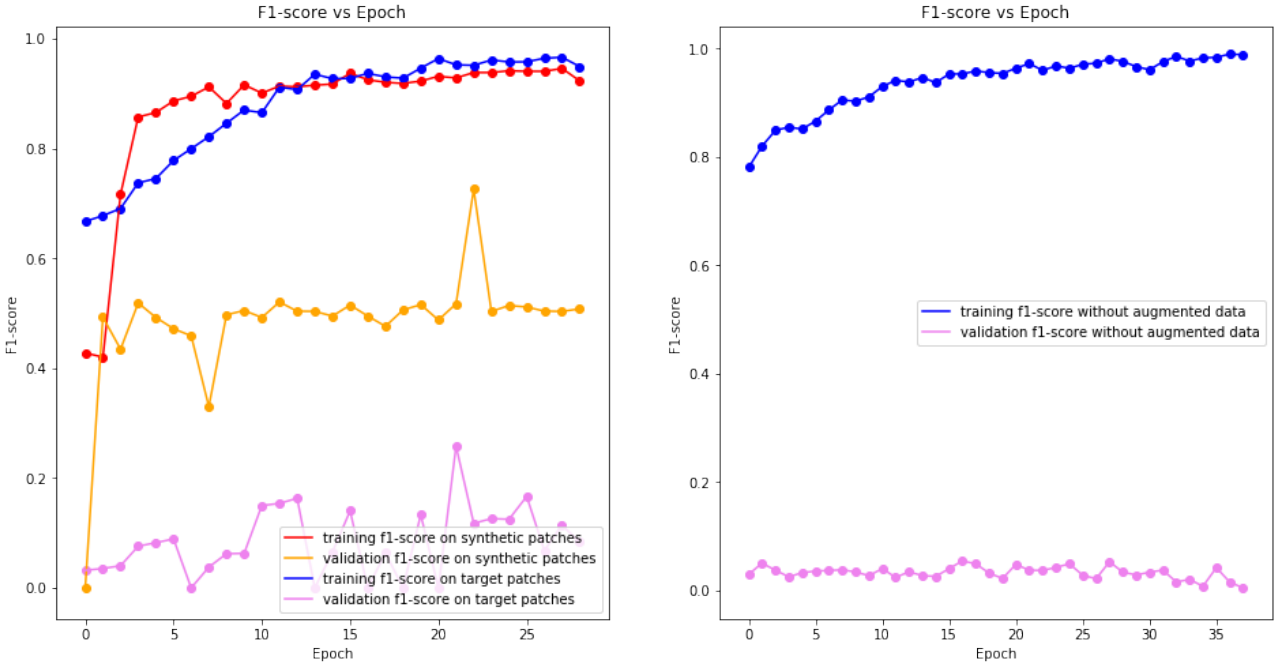

Figure 6: Train & Validation F1 score

3.2. Testing

For testing we have taken the model with the highest average IOU over the validation data.

It can seen from the figure 4 that the model is over-fitting on the training data even after training with synthetic data.

Average IOU on test data	
Without Augmentation	0.00138
With Augmentation	0.00178

3.3. Examples

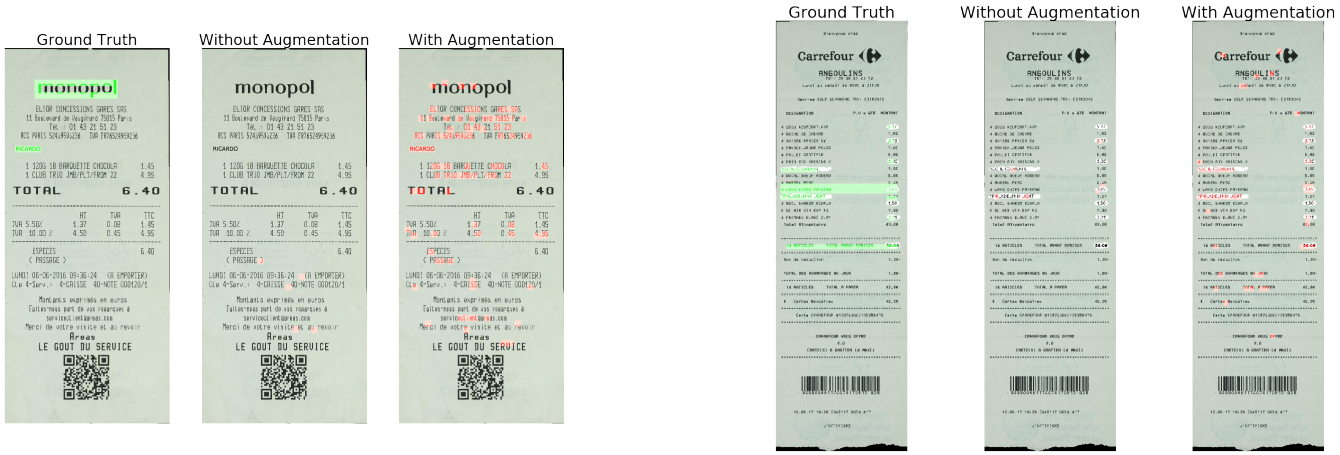

Figure 7: Results of detection on test data

For more results go to this link

4. Conclusion

We can see that the model is performing very poorly even after training with synthetic images. A large amount of mis-classification is resulting in very low precision and f1-score. A change in architecture to models like RCNN should definitely provide better results.

References

[1] *C. Artaud, N. Sidre, A. Doucet, J. Ogier, V. P. D. Yooz, Find it! fraud detection contest report, in: 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 13–18.* doi:10.1109/ICPR.2018.8545428.

[2] *Y. Annadani, C. V. Jawahar, Augment and adapt: A simple approach to image tampering detection, in: 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 2983–2988.* doi:10.1109/ICPR.2018.8545614.

[3] *Z. Tian, W. Huang, T. He, P. He, Y. Qiao, Detecting text in natural image with connectionist text proposal network (2016).* arXiv:1609.03605.