Fake Currency Note Identification using Deep Convolutional Neural Networks

Hritik Jain
IIT(BHU), Varanasi
Email: hritik.jain.cse13@iitbhu.ac.in

S.K. Pal
DRDO, New Delhi
Email: skptech@yahoo.com

***Abstract*—Circulation of counterfeit currency poses a serious threat to the national economy, financial institutions and consumers worldwide. Previous approaches to identifying fake currency notes from images rely heavily on image processing techniques. A new approach towards identification of fake currency notes through their images is examined in this paper. The method is based on Deep Learning, which has seen tremendous success in image classification tasks in recent times. An artificial data-set was generated to assist training and testing of the deep convolutional neural network (DCNN) model. Preliminary results from the experiments are very encouraging and demand further research. Under the availability of a real data-set of images of fake currency notes, the DCNN can be trained more robustly. Through improvements in architecture of the DCNN, the technique can be adapted into a smart-phone app which will help people in identifying a fake currency note in real time through a camera image of the same.**

Keywords*—*fake currency, deep learning, convolutional neural network, fine-grained image classification, transfer learning.

I. Introduction

The recent demonetisation drive may be a step towards eradication of corruption and black money, but it fails to address the problem of counterfeit(fake) currency. A counterfeit note of INR 2,000 denomination was traced just two days after the notes went into circulation on November 8, 2016. According to a study conducted by the Indian Statistical Institute prior to the demonetisation drive, as many as 250 out of every 10 lakh notes in circulation were fake.

Knowingly possessing fake currency notes is a punishable offense under the IPC section 489C and can lead to imprisonment ranging from seven years to even life imprisonment depending upon the crime. The Crime Investigation Department (CID) believes that it is possible to detect fake currency, even though it is printed with precision. Currency printed by local racketeers and foreign racketeers can be detected easily as they use the photographic method, hand engraved blocks, lithographic processes and colour scanning. The watermark is made by using opaque ink, painting with white solution, stamping with a dye engraved with the picture of Mahatma Gandhi. Oil, grease or wax is then applied to give the picture a translucent feel. The security thread in genuine notes is incorporated into the paper at the time of manufacture. But in fake notes, the security thread is imitated by drawing a line with a pencil, by printing a line with grey ink, or by using aluminium thread while pasting two thin sheets of paper. It becomes difficult for the forgers to reproduce the same shape of individual numbers consistently with accuracy. Maintaining alignment of figures is also a challenge. The printed lines will be broken and there may also be ink smudges.

Identifying a fake currency note by mere visual inspection is still a very difficult task. An average individual is not fully aware of all the security features present in a currency note and thus remains vulnerable to fraud. Equipping smart-phones with applications which can identify a currency note as fake through a camera image [3] is one promising direction towards solving this problem.

Deep learning models have seen tremendous success in image classification tasks [25]. And identifying a currency note as fake or real from its image is essentially a binary image classification task. In this paper, we test the feasibility of deep learning models for fake currency identification, which can be trained without manual feature extraction [1,2,4,5,6,7] on raw images of currency notes. We will later note that such a model can be built into a smart-phone app for real-time fake-currency detection through the smart-phone camera.

The rest of the paper is structured as follows. Section II observes previous approaches to this problem. In Section III, we formulate the problem of fake currency identification as a case of binary *fine-grained image classification*. In Section IV, we describe our approach to the problem, starting with generation of a data-set, the architecture of the CNN used in our experiments and followed by the results of the experiments. Conclusions and possible future avenues of research are given in Section V.

II. Related Work

Previous approaches to this problem [1,2,4,5,6,7,9] have been based on image-processing techniques, where features of the currency note were manually extracted. After the acquisition of image, image pre-processing is performed. This includes smoothening or noise removal of the image, grayscale conversion, thresholding etc. After the pre-processing step, any of the various edge detection techniques are applied to find the boundary of a currency note and the cropped image is passed on for further analysis. Image segmentation process can then be applied [2,4,6,9] to partition the image into multiple regions, to locate objects and boundaries (lines, curves, etc.) and extract the ROIs (Region of Interest) using cropping. Next, feature extraction is performed on the extracted pre-processed areas. This may involve extracting the H,S,V components of particular blocks (ROIs) [2,6] in the image of a note. This way, unique and distinguishing features are found under various challenging conditions such as different illumination

and background, old and worn notes. The extracted feature values are compared with ideal(known) feature values of a real note and the note is accordingly classified as fake or real.

III. Proposed Solution

A. Fine-grained Image Classification

Classifying a currency note as fake or real can be considered to be a case of fine-grained image classification, i.e., classification among categories which are both visually and semantically very similar. This is a very difficult regime which is even challenging for humans without careful training. Examples include fine distinction into species of animals and plants, of car and motorcycle models, of architectural styles, etc. [11]

B. Transfer Learning

For image classification tasks which are concerned with very specific kind of images, images of currency notes in our case for instance, a deep convolutional neural network (DCNN) is not trained from scratch in practice. This is because it is relatively rare to have a data-set of sufficient size that is required for the depth of the convolutional neural network required. Instead, it is common use the weights of a DCNN pre-trained on a very large data-set and fine-tune the weights of the pre-trained network by continuing the back-propagation on new data-set. This methodology is known as transfer learning [26].

Now, for the purpose of fine-tuning the weights, we have a choice. One can choose to either fine-tune all the layers of the DCNN, or keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a DCNN contain more generic features (e.g. presence of edges or color patches) that should be useful to many tasks, but later layers of the DCNN become progressively more specific to the details of the classes contained in the new data-set.

Size of the new data-set and its similarity to the original data-set also influences the transfer learning strategy to be used for a given image classification task. With the small data-set that we generate (Section IV-A), which is also very different to the original data-set that the DCNN we use is pre-trained on (Section IV-B), we have the following points to consider.

- It is not advisable to fine-tune the entire DCNN due to overfitting concerns.
- It is likely best to only train a linear classifier on the CNN-features.
- It might not be best to train the classifier from the top of the network, which contains more data-set specific features. Instead, training a classifier from activations somewhere earlier in the network might be a better choice.

We now look at some similar image-classification tasks where the above two concepts were used. In [12], a general DCNN model was fine-tuned for the task of plant classification which included distinguishing herb, tree and fern species. The base model that was used is the best performing model from ILSVRC 2014 [22], referred to as GoogLeNet. However, the DCNN was only used as a feature extractor and not as a classifier. The features were then used to learn a multi-class linear SVM. Similarly in [16], DCNNs were trained to extract features and the final classification was performed by a linear SVM. In [17], an extremely randomised forest was used to classify the generic pre-trained features. The local features extracted by deep convolutional neural networks were modelled to improve fine-grained image classification in [15].

It is therefore evident that we can rely on DCNNs to work as feature extractors. We want to leverage this capability to learn the subtle security features of real currency notes without involving any manual image processing methods. In our experiments however, we use a softmax layer on top of the DCNN so that it works as a classifier itself. Further, to keep the experiments simple and get preliminary results, we decided to fine-tune the entire DCNN with our small data-set (Section IV-A). In view of the concerns raised under Transfer Learning (Section III-B), this decision may lead to over-fitting on the data-set. We hope that under availability of a much larger data-set of real and fake notes, the DCNN can more accurately learn the security features and generalize better at the same time.

IV. Our Approach

A. Data-set Generation

The new 2000 rupee note is loaded with 17 security features. Out of these, 2 features for the visually impaired are not useful for our model which is to be trained completely on the images of the primary face (bearing portrait of Mahatma Gandhi) of currency notes. In addition, 4 of these features are located on secondary face on the note but are helpful for our data-set generation as we use images of currency notes held against light to expose most of the security features. Similar features with slight variations in terms of figures and relative locations are present on the new 500 rupee note.

We generated two different data-sets:

- **New 2000 rupee note data-set:**
  Initially, there were 3 smart-phone camera images each of a real new 2000 rupee note and the photocopy of a 2000 rupee note (to simulate fake notes) which was destroyed immediately after clicking the images. The images were taken by holding the note and its photocopy against light. This is a requirement as we want all the security features in a real currency note exposed, which involve features like the watermark of Mahatma Gandhi and the see-through register. In every image, the currency note covered approximately the entire frame of the image. This provided us with a total of 6 initial images in the data-set.
- **New 500 rupee note data-set:**
  To start with, there were 4 smart-phone camera images of 4 real new 500 rupee notes. Again, the images were taken by holding the note against light. To simulate fake 500 rupee note images, we manually erased the exposed watermark through a basic photo editing software. The motive behind this was to train the

(a) Real 2000 rupee note

(b) Photocopy of 2000 rupee note

(c) Real 500 rupee note

(d) Fake 500 rupee note

Fig. 1. Initial images for generating the data-sets.

DCNN to learn watermark as the sole security feature of a real currency note for experimental purposes. But unlike the case of the 2000 rupee note data-set, the images were taken such that the 500 rupee notes did not cover the entire frame of the image and covered only around 60% central area of the image. We will see in the next section how this affected the results. Next, we also applied some visual effects on these images using online photo editing tools to simulate old, worn, scribbled notes and notes with spilled color. This provided us with a total of 26 initial images to create this data-set.

We required at least 1000 images in each data-set even for the purpose of fine-tuning a pre-trained DCNN. The simplest way to get around a lack of data is to augment the data-set. The principle behind data augmentation is that from already qualified data, we can generate more by modifying our base. The main benefit or data augmentation is that we get more data, and our training and/or testing set gets better. The second one is that since were adding noise, we get closer to real data, and can improve the measurement of a real-world behavior of our machine learning technique. For data augmentation purposes, we made use of the OpenCV library [19]. The very first step is to resize the image to a pre-defined size. The VGG-16 model that we use in our experiments (see next section) requires an input image shape of 224 x 224 x 3, where 3 refers to the R,G,B components of a coloured image and the image must be 224 x 224 pixels in size. We then apply the following three types of transformations for each original image in both the data-sets to augment them.

1) **Image Translation:** Translated each image with a stride of 5 pixels along both axis along all 4 directions. The range for translation was -30 pixels to +30 pixels. The original image moves withing the frame and black pixels fill the void created thereby.
2) **Image Rotation:** Rotated each image in the angle range of -10 to +10 degrees with a step equal to 0.5 degree. The axis for rotation passes through the centre of the image and black pixels fill the void created by rotation.
3) **Perspective Transformations:** Applied perspective transformations on each image to zoom in in the range of 100-130% and zoom out in the range of 100-70%. This added a total of 60 images for each original image in the data-set.

The data augmentation process generated 270 images for each original image, resulting in a total of 1620 images for the *New 2000 rupee note data-set* and 7020 images for the *New 500 rupee note data-set.*

B. Architecture of the DCNN

For our experiments, we have used the python libraries Lasagne and Theano [21] for implementing and training the deep learning model. There are many pre-trained models available on the internet like the GoogleNet [22], ResNet [23], InceptionV3 [24], VGG [20], etc. The VGG networks secured first and the second places in the localisation and classification tracks respectively in the ImageNet Challenge [27] 2014. We chose the VGG model for our experiments because it has a very simple structure (see Fig. 2) compared to the other top performing DCNN models. This makes it easier to interpret. Out of the two best-performing VGG models publicly available to facilitate research, we use the VGG-16 model, the other being the VGG-19 model. In Fig. 2, VGG-16 is marked by the green rectangle. It is called VGG-16 because it consists of 16 layers with learnable parameters, i.e. weights and biases. The pooling layers dont count as they do not learn anything. To reduce the number of parameters in such very deep networks, small 3x3 filters are used in all convolutional layers with the convolution stride set to 1. At the end of the network are three fully-connected (or fc) layers. The VGG networks use multiple 3x3 convolutional layers to represent complex features. A downside of the VGG model is that it uses a lot of memory and parameters (140M). Since the original VGG model was designed to classify among 1000 classes for the ImageNet Challenge, the final 'fc' layer has 1000 units. We

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 2. VGG deep convolutional neural network architecture from the original paper.

replace this layer by a 'fc' layer with 2 units, as we have only two classes to distinguish between. This layer applies softmax as the non-linearity to output probabilities of the two classes, fake and real.

C. Training the DCNN

The model described above was trained using the two data-sets separately. Each of the data-sets was randomly divided into training and test sets. The ratio for this division was 2:1. The model was trained for 5 epochs on the New 2000 rupee note data-set and for 10 epochs on the New 500 rupee note data-set. Training was done on the Nvidia 645M GPU which has a CUDA Compute Capability of 3.0. Training took around 2 hours for the New 2000 rupee note data-set and around 12 hours for New 500 rupee note data-set.

D. Results

The accuracy of the VGG-16 model fine-tuned with the New 2000 rupee note training set was 100% on the corresponding test set. Despite the perfect score, the result might be attributed to overfitting on the data-set, as the training and test sets were very similar. This result is still very encouraging because an availability of a larger, real-world data-set can make the model more accurately trained and generalize better at the same time. The accuracy of the VGG-16 model fine-tuned with the New 500 rupee note data-set was 66% on the corresponding test set. This result is not encouraging but can be attributed to the fact that the New 500 rupee note data-set had images which only covered around 60% of the central area of the image and shrinked further to 224 x 224 pixels as per the VGG-16 architecture requirement. The task of applying image processing techniques to first detect the edges of a currency note in an image and passing the cropped image to the model can be taken up in future.

V. Conclusion

In this paper, we have demonstrated the feasibility of deploying deep learning techniques for the task of identifying fake currency notes, using the VGG-16 deep convolutional neural network model. We noted how deep convolutional neural networks can work as feature extractors and thus no image processing techniques need to be applied to manually find the presence of security features in a note. Although the generated data-set did not represent the real-world scenario of fake currency notes, it was helpful for experiments. Under the availability of a real data-set, the deep neural networks can be better trained. Such a model may then be built into a smart-phone app and can thus help people in detecting fake notes in case of suspicion in real time with just an image taken through the smart-phones camera. In the form of a smart-phone app, the VGG-16 model has been reported to take up to 550 MB of device memory. In a phone with a with a high-end processor, initialization of the app takes anywhere between 2 to 10 seconds and classification time is around 0.25 to 0.3 seconds per image.

Future avenues of research include examining various deep neural network architectures which are more efficient in terms of time and space complexity. Applying image pre-processing techniques like noise removal and edge-detection to crop the currency note out of an image will present a better input to the deep neural network.

REFERENCES

[1] Renuka Nagpure, Shreya Shetty, Trupti Ghotkar, Chirayu Yadav, Suraj Kanojiya, "Currency Recognition and Fake Note Detection", International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, Issue 3, March 2016.

[2] Rinki Rathee, "Design of HSV Mechanism for Detection of Fake Currency", International Journal of Emerging Technology and Advanced Engineering, vol. 6, Issue 7, July 2016.

[3] D.Alekhya , G.DeviSuryaPrabha , G.VenkataDurgaRao, "Fake Currency Detection Using Image Processing and Other Standard Methods", International Journal of Research in Computer and Communication Technology, vol. 3, Issue 1, January 2014.

[4] M.Deborah, C.Soniya Prathap, "Detection of Fake currency using Image Processing", International Journal of Innovative Science, Engineering & Technology, vol. 1 Issue 10, December 2014.

[5] S.Atchaya, K.Harini, G.Kaviarasi, B.Swathi ,"Fake Currency Detection Using Image Processing", International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Special Issue.

[6] Binod Prasad Yadav, C. S. Patil, R. R. Karhe, P.H Patil, "An automatic recognition of fake Indian paper currency note using MATLAB", International Journal of Engineering Science and Innovative Technology (IJESIT), vol. 3, Issue 4, July 2014.

[7] Swami Gururaj M., Naveen J.,"Identification of Counterfeit currency and denomination using Raspberry pi", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 4, Special Issue 2, April 2016.

[8] Trisha Chakraborty, Nikita Nalawade, Abhishri Manjre, Akanksha Sarawgi, Pranali P Chaudhari, "Review of Various Image Processing Techniques for Currency Note Authentication", vol. 3, pp. 119-122, Issue 3, March 2016.

[9] Eshita Pilania, Bhavika Arora, "Recognition of Fake Currency Based on Security Thread Feature of Currency", International Journal Of Engineering And Computer Science, vol. 5, pp. 17136-17140, Issue 7, July 2016.

[10] P. Julia Grace, A. Sheema, "A Survey on Fake Indian Paper Currency Identification System", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, Issue 7, July 2016.

[11] Zhang, Ning, Jeff Donahue, Ross Girshick, and Trevor Darrell, "Part-based R-CNNs for fine-grained category detection", European Conference on Computer Vision, pp. 834-849, Springer International Publishing, 2014.

[12] Ge, ZongYuan, Chris McCool, Conrad Sanderson, and Peter Corke, "Content Specific Feature Learning for Fine-Grained Plant Classification", In Conference and Labs of the Evaluation Forum (Working Notes), 2015.

[13] Jiang Wang et al., "Learning fine-grained image similarity with deep ranking", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014.

[14] Kunze, Ike Sebastian, "Fine-Grained Image Classification with Convolutional Neural Networks", Hauptseminar im Wintersemester 2014/15 Medizinische Bildverarbeitung, p.55.

[15] Z. Ge, C. McCool, C. Sanderson, and P.I. Corke., "Modelling local deep convolutional neural network features to improve fine-grained image classification", IEEE International Conference on Image Processing, 2015.

[16] ZongYuan Ge, Alex Bewley, Christopher McCool, Ben Upcroft, Conrad Sanderson, and Peter Corke, "Fine-Grained Classification via Mixture of Deep Convolutional Neural Networks", IEEE Winter Conference on Applications of Computer Vision, 2016.

[17] Niko Sunderhauf, Chris McCool, Ben Upcroft, and Tristan Perez, "Fine-Grained Plant Classification Using Convolutional Neural Networks for Feature Extraction", In Conference and Labs of the Evaluation Forum (Working Notes), pp. 756-762, 2014.

[18] Marion Chevalier, Nicolas Thome, Matthieu Cord, Jerome Fournier, Gilles Henaff, and Elodie Dusch, "LR-CNN for fine-grained classification with varying resolution", International Conference on Machine Learning, 2015.

[19] Itseez, Open Source Computer Vision Library, https://github.com/itseez/opencv, 2015.

[20] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", Proceedings of International Conference on Learning Representations, 2014.

[21] Y. Bengio et al., "Theano: A CPU and GPU Math Expression Compiler", Proceedings of the Python for Scientific Computing Conference (SciPy) 2010, June 30 - July 3.

[22] Christian Szegedy et al. "Going deeper with convolutions.", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna,"Rethinking the inception architecture for computer vision", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[25] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks", In Advances in neural information processing systems, pp. 1097-1105, 2012.

[26] Appears in the Handbook of Research on Machine Learning Applications, published by IGI Global, edited by E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, 2009.

[27] Russakovsky, Olga, et al., "Imagenet large scale visual recognition challenge", International Journal of Computer Vision, 115(3), pp.211-252.