ANALYSIS ON CAPSULE NETWORKS TO DETECT FORGED IMAGES AND VIDEOS

Pasunuri Neha Reddy[1], Aruva Vaishnavi[2], Ashutosh Mengji[3], G Mukesh[4]

1,2,3: Department of electronics and communication engineering, Sphoorty engineering college, Hyderabad, India.

4 Head of the department, asst professor, department of electronics and communication engineering, sphoorty engineering college, Hyderabad, India.

ABSTRACT

Modern day computer vision tasks requires efficient solution to problems such as image recognition, natural language processing, object detection, object segmentation and language translation. Symbolic Artificial Intelligence with its hard coding rules is incapable of solving these complex problems resulting in the introduction of Deep Learning (DL) models such as Recurrent Neural Networks and Convolutional Neural Networks (CNN). However, CNNs require lots of training data and are incapable of recognizing pose and deformation of objects leading to the introduction of Capsule Networks. And these are the new sensation in Deep Learning. They have lived to this expectation as their performance in relation to the above problems has been better than Convolutional Neural Networks. Even with this promise in performance, lack of architectural knowledge and inner workings of Capsules serves as a hindrance for researchers to take full advantage of this breakthrough. In this paper, we provide a comprehensive review of the state of the art architectures, tools and methodologies in existing implementations of capsule networks. We highlight the successes, failures and opportunities for further research to serve as a motivation to researchers and industry players to exploit the full potential of this new field. The main contribution of this survey article is that it explains and summarizes significant current state of the art Capsule Network architectures and implementations. The method introduced in this paper uses a capsule network to detect various kinds of spoofs, from replay attacks using printed images or recorded videos to computer generated videos using deep convolutional neural networks. It extends the application of capsule networks beyond their original intention to the solving of inverse graphics problems.

1. INTRODUCTION

Recent advances in media generation techniques have made it easier for attackers to create forged images and videos. State of-the-art methods enable the real-time creation of a forged version of a single video obtained from a social network. Although numerous methods have been developed for detecting forged images and videos, they are generally targeted at certain domains and quickly become obsolete as new kinds of attacks appear. The revolution in computer hardware, especially in graphics

processing units and tensor processing units, has enabled significant advances in computer graphics and artificial intelligence algorithms. In addition to their many beneficial applications in daily life and business, computer-generated/manipulated images and videos can be used for malicious purposes that violate security systems, privacy, and social trust. The deepfake phenomenon and its variations enable a normal user to use his or her personal computer to easily create fake videos of anybody from a short real online video. Several countermeasures have been introduced to deal with attacks using such videos.

Although numerous methods have been developed for detecting forged images and videos, they are generally targeted at certain domains and quickly become obsolete as new kinds of attacks appear. Capsule Networks (CapsNet) are the networks that are able to fetch spatial information and more important features so as to overcome the loss of information that is seen in pooling operations. So capsule networks are effective towards each attack which decreases the cyberattacks. Even though, the face manipulation is not new, recent achievements demonstrate that computer-manipulated faces can reach a photo-realistic level at which it is almost impossible for them to be humanly detected as fake. The introduction of deep learning significantly improved the ability of this kind of attack and thus attracted great attention from the forensics community. They argued that CNNs have limited applicability to the "inverse graphics" problem and introduced a more robust architecture comprising several "capsules."

Several countermeasures have been proposed to deal with manipulated images and videos. However, most of them are aimed at particular types of attacks. For example, local binary pattern (LBP)-based methods are effective against replay attacks in which the attacker places a printed photo or displays a video on a screen in front of the camera. However, the eyes-focused method designed to detect a deep fake forgery can fail with the replay attack when the video displayed is of the actual target person. Other methods have more generalized ability; for instance, Fridrich and Kodovsky's method can be applied for both steganalysis and detecting facial reenactment videos. However, its performance on secondary tasks is limited in comparison with task-specific methods like that of Rossler et al. Moreover, while some methods can detect a single forged image, others require video input. This paper presents a method that uses a capsule network to detect forged images and videos in a wide range of forgery scenarios, including replay attack detection and (both fully and partially) computer-generated image/video detection. This is pioneering work in the use of capsule networks, which were originally designed for computer vision problems, to solve digital forensics problems. A comprehensive survey of state-of-the-art related work and intensive comparisons using four major datasets demonstrated the superior performance of the proposed method.

However, they initially faced the same problem faced by CNNs – the limited performance of hardware and the lack of effective algorithms, which prevented practical application of capsule networks. CNNs thus remained dominant in this research field. These problems were overcome when the dynamic routing algorithm and its variance – the expectation-maximization routing algorithm were

introduced. However, most of them are targeted at certain domains and are ineffective when applied to other domains or new attacks. In this paper, we introduce a capsule network that can detect various kinds of attacks, from presentation attacks using printed images and replayed videos to attacks using fake videos created using deep learning. It uses many fewer parameters than traditional convolutional neural networks with similar performance. Moreover, we explain, for the first time ever in the literature, the theory behind the application of capsule networks to the forensics problem through detailed analysis and visualization. The proposed method works for both images and videos. For video input, the video is split into frames in the pre-processing phase. The classification results (posterior probabilities) are then acquired from the frames. The probabilities are averaged in the post-processing phase to get the final result. The remaining parts are constructed the same way as when the input is an image. In the pre-processing phase, faces are detected and scaled to 128/128. This project helps to detect forged images and videos using capsule networks and dynamic routing algorithm. Now a days the issue of forgery has taken over the world. Currently detecting forged images and videos is limited to certain domains. The motive is to detect forged images and videos irrespective of the domain and the attacker.

Software Requirements

For developing the application, the following are the Software Requirements:

1. Python
2. Django
3. MySQL
4. Wamp server.

Hardware Requirements

For developing the application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 2 GB
- Space on Hard Disk: Minimum 512MB.

Other Requirements

Functional requirements will define the fundamental actions that must take place in the software in accepting & processing the inputs in processing & generating the outputs. Graphical User interface with the User is required. Operating Systems supported is Windows 7 and higher. Information flows is Class diagram, Uses Case Diagram, Sequence Diagrams, Activity Diagrams will be provided which describes the flow of data between various processes of the system. The Debugger and Emulator is Any Browser (Particularly Chrome).

2. LITERATURE SURVEY

The basic structure of the first successful capsule network was that of Sabour et al., [1] (2017), which consisted of two convolution layers. Conv1 had 256 channels each made up of 9×9 filters with a stride of 1 and a ReLU activation function applied to a $28 \times 28 \times 1$ MNIST image.The second layer was designed as a convolutional capsule layer with $6 \times 6 \times 32$ capsules each outputting an 8D vector. At a stride of 2, each primary capsule has 8 convolutional units operating with a 9×9 kernel. Non-linearity is obtained by applying a squashing function (as activation function) to create 10, 16D capsules. This layer is also the primary capsule layer receiving features as scaler outputs from Conv1 layer after which all layers will have to deal with 8D vector values. The layer consists of 32 channels each with 6×6 grid of primary capsules. The third layer (DigitCaps) is a fully connected layer with ten 16D capsules, each receiving input from all capsules from the layer below to perform classification based on 10 classes. The last layer determines the length of each capsule in the previous layer necessary to obtain the probability that the entity is present. Reconstruction of the image is done using the decoder made up of fully connected (FC) layers (as shown in Figure 2.1.).

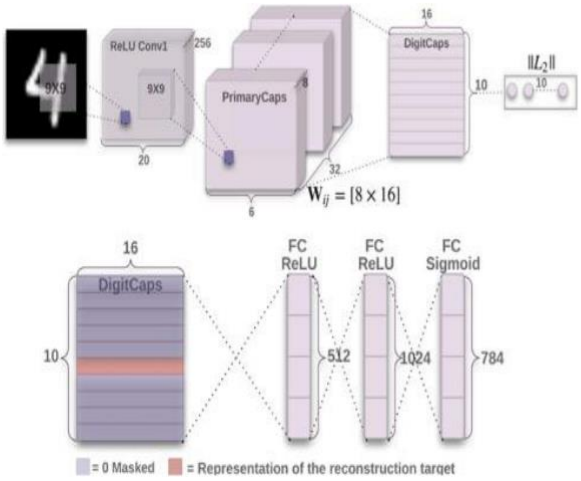


Figure 2.1: Structure of the capsule network

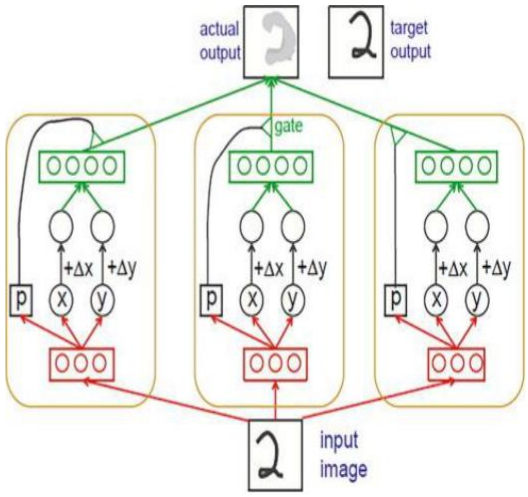


Figure 2.2: Auto-encoder Capsule structure

Hinton et al., [2] (2011) addressed the limitations of CNNs applied to inverse graphics tasks and laid the foundation for a more robust "capsule" architecture in 2011. However, this complex architecture could not be effectively implemented at the time due to the lack of an efficient algorithm and the limitations of computer hardware. Instead, easy-to-design easy-to train CNNs became widely used. Now, with the introduction of the dynamic routing algorithm and the expectation maximization routing algorithm, capsule networks have been implemented with remarkable initial results. Two recent studies demonstrated that, with the agreement between capsules calculated by the dynamic routing algorithm, the hierarchical pose relationships between object parts can be well. To explain the source of the first-level capsules and how an ANN can learn to convert pixel intensities to pose parameters, a 2D image is used as depicted in the figure 2.2.

Zhang et al., [3] (2018) did implementation of CapsNet has seen several proposed modifications in a bid to resolve some limitations of the original implementation and improve performance. Critical analysis of the routing-by-agreement shows it does not automatically ensure a higher-level capsule (parent) is coupled with multiple lower level capsules (children) to form a parse tree. Instead of allowing lower level capsules to send their outputs to all higher-level capsules as is the case in the original routing-by-agreement algorithm, a lower level capsule can select a single parent enabling the network to be deep and resilient to Whitebox adversarial attacks. In this light, a generative adversarial network (GAN) with a highly performing discriminator made from capsules can be helpful in determining whether a given image is natural or artificially created (fake). The better the discriminator (critic) is, the faster the generator learns. However, modelling the probability distributions of data in deep generative models soon becomes intractable. As a result, CapsNets become better alternatives to CNNs as discriminators in a GAN, ensuring that vital information is not lost through pooling. The dynamic routing algorithm can be seen as an optimization problem that can be formulated as a minimization of an objective function. This modification prevents the activation probability from becoming highly unbalanced as the number of iterations increases.

Justus et al., [4] (2016) to stabilize the training process, Sabour [1] performed regularization on the weight matrix by using a margin loss. A more general solution is to rescale the weight matrix and ensure that the inner product between the input and the weighted sum (sj) of all the individual primary capsule predictions for capsule j is set below 1 for each iteration. Equal weight initialization routing scheme in the original CapsNet has the tendency to slow convergence and result in poor accuracy. A better option for faster convergence and improved accuracy is to model the initial routing weights as trainable parameters to be trained by back propagation. We can leverage on the realization (and using the fact) that primary capsule predictions are not independent to improve the performance of CapsNets on multi-label classification tasks. Focusing on the length of a capsule rather than individual capsule outputs proves to be a better entity detection approach. In this case, the lengths of capsules rank an entity's presence and their orientation instantiates its pose (position, size, and orientation), deformation, velocity, albedo, hue, texture and so on. In the Softmax function is used to normalize routing coefficients representing assignment probabilities between capsules of adjacent layers. However, Softmax constraints the possible set of values the routing coefficients can assume, leading to uniform probabilities after several routing iterations.

Kim et al., [5] (2018) the loss in performance from this problem can be addressed by using Max-Min function in place of Softmax. Max-Min performs scale invariant normalization allowing lower level capsules to take on independent values as opposed to what Softmax does. To learn discriminative feature maps better for subsequent use by CapsNets, convolutional layers can be replaced with densely connected convolutional layers. However, increasing network depth results in vanishing gradient problems. Fortunately, it can be resolved by feeding subsequent layers with signals from previous layers of ResNet

Cohen et al., [6] (2017) and Chung et al., [7] (2017) or by adding dense connections between every layer in a feed forward manner Seitz et al., [8] (2017) with feature concatenations. It can also be argued that the routing procedure in CapsNets is not properly integrated into the training procedure since the routing procedures are not embedded into the optimization procedures and the choice of the optimal number of routing must be found manually Chingovska et al. [9] (2012).

This manual selection does not guarantee convergence. Capsules can perform comparably better than CNN Transfer learning models (e.g. InceptionNet and DenseNet) on other datasets aside MNIST or small NORB Pereira et al., [10] (2013) with additional configurations such as increased number of Conv layers and FC layers Li et al., [11] (2018). Dynamic routing based CapsNets do not guarantee equivariance or invariance Kodovsky et al.,
[12] (2012). This is due to the fact that space of a pose is a manifold whiles votes are averaged in a vector space failing to produce equivariance mean estimates in the manifold. In addition, trainable transformation kernels in the capsule layers are defined over local receptive fields found in the spatial vector domain which has receptive field coordinates that are ignorant of the pose. Nguyen et al., [13] (2018) proposed group equivariant capsules layers with pose vectors restricted as elements of a group. Under a general routing by agreement algorithm and under certain conditions, equivariance and invariance can be guaranteed for such groups.

3. SYSTEM DESIGN

Introduction: The Unified Modelling Language (UML) is a visual modelling language used to specify, visualise, construct and document a software intensive system. The embedded real-time software systems encountered in applications such as telecommunications, school systems, aerospace, and defence typically tend to be large and extremely complex. It is crucial in such systems that the software is designed with a sound architecture. A good architecture not only simplifies construction of the initial system, but also readily accommodates changes forced by a steady stream of new requirements. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. The primary goals in the design of the UML are: Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models. Provide extensibility and specialisation mechanisms to extend the core concepts. Be independent of particular programming languages and development processes. Provide a formal basis for understanding the modelling language.

Encourage the growth of the OO tools market. Support higher-level development concepts such as collaborations, frameworks, patterns and components. Integrate best practices.

System Architecture

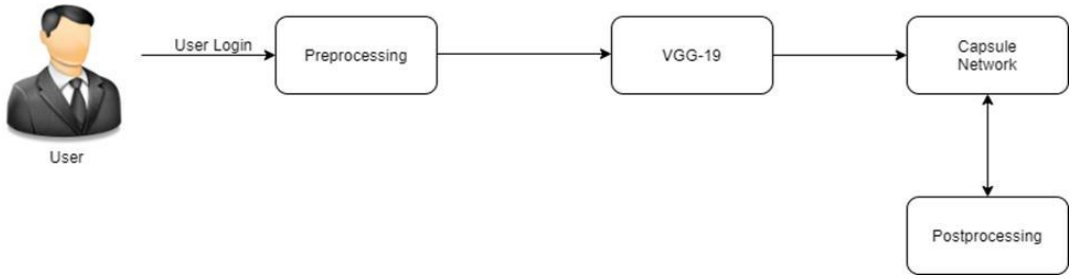


Figure 3.1: system architecture

System Flow

Design is a meaningful engineering representation of something that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where quality is fostered in software engineering. Design is the perfect way to accurately translate a customer's requirement into a finished software product. Design creates a representation or model, providing detail about software data structure, architecture, interfaces and components that are necessary to implement a system. This chapter discusses the design part of the project. Here in this document the various UML diagrams that are used for the implementation of the project are discussed.

Modelling: The UML Class shows the static structure of the model. The class is a collection of static modelling elements, such as classes and their relationships, connected as a graph to each other and to their contents. The figure 3.2 shows the activity diagram.

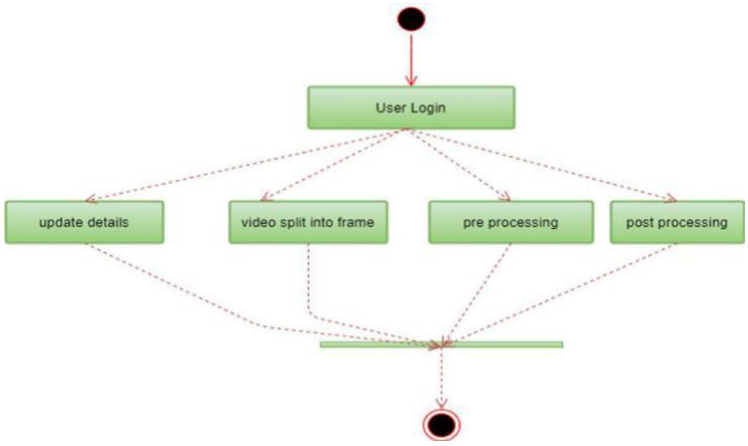
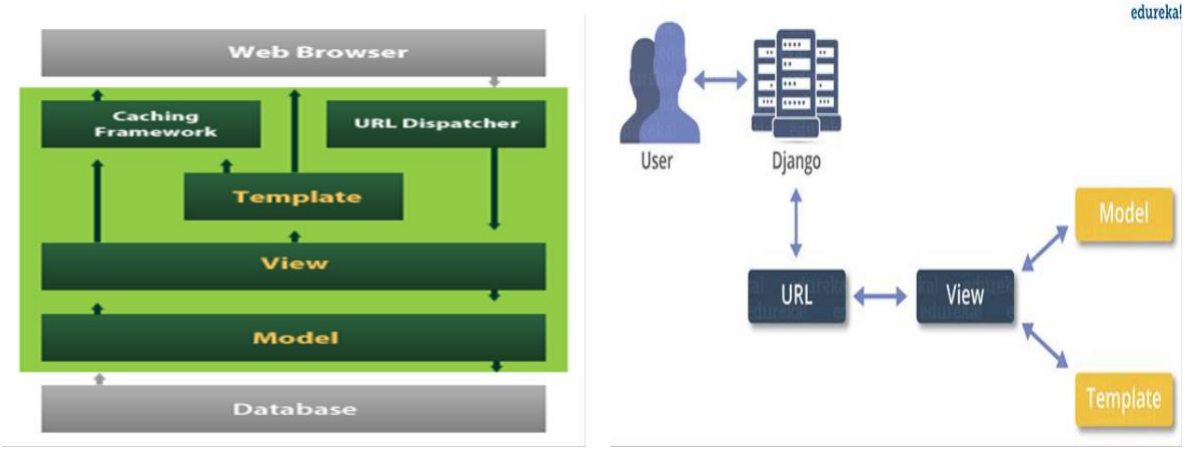


Figure 3.2: Activity diagram

4. IMPLEMENTATION

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models as shown in figure 4.1.



4.1: Django work flow

Django is a web application framework written in Python programming language. It is based on the MVT (Model View Template) design pattern. Django is very demanding due to its rapid development feature. It takes less time to build an application after collecting client requirements. This framework uses a famous tagline: The web framework for perfectionists with deadlines. By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of the configuration automatically, so we can focus on application development only.

Implementation of Modules

The implementation of the project is divided into six sections. In the first section, we are going to import the required libraries and then study them. Next, we are going to prepare the dataset with required attributes, then transformations on data are performed, and then data analysis can be made using correlation, followed by splitting of a dataset into train and test sets, finally, model training is done to know the best model that fit(s) our data for predicting rainfall rate.

Step-1: Import Libraries:

tensorflow==1.14.0

keras==2.1.2

Step-2: Prepare Dataset

In a test to detect computer-manipulated images and videos, the database we used has two parts

Real images:

/dataset/<train;test;validation>/Real

Fake images:

/dataset/<train;test;validation>/Fake

The dataset need to be pre-processed to crop facial area.

Step-3: Data Preprocessing

Data Preprocessing is the most vital step while preparing our dataset for model training. Data is often inconsistent, incomplete, and consists of noise or unwanted data. So, preprocessing is required. It involves certain steps like handling missing values, handling outliers, encoding techniques, and scaling. Removing null values is most important because the presence of null values will disturb the distribution of data, and may lead to false predictions. There is very less percent of null values in the dataset.

Step-4: Visualization using the technique of Correlation

A common visualization technique is to show the reconstructed images to enable visual comparison with the original ones. These reconstructions can either be done at the end of selected epochs, at the end of the training or during testing. A plot of accuracy or loss or mean average precision (*maP*) against the number of epochs is also common in literature. There are also rare cases of precision against recall plots especially when using capsules for Natural Language Processing (NLP) tasks and loss against number of batches. Heat maps are also common in literature. Statistical forms of visualization such as histograms, curve plots, tables and their likes are powerful tools that are sometimes used to report the performance of a model. The objectives of these visualizations are twofold: first to help us use our human intuition based on vision in evaluating how good the performance of the algorithm is. Secondly, to provide us with some level of understanding of the internal operations of the model.

Step-5: Model Training

The pre-processed image then passes through a part of the VGG-19 network (as proposed by the Visual Geometry Group) pre-trained on the ILSVRC database before entering the capsule network. The VGG-19 network is used from the first layer to the third max pooling layer, which is not too deep to obtain biases from the object detection task (the original purpose of this pre-trained network). This VGG-19 part is equivalent to the CNN part before the primary capsules in the design of the original capsule network. Using a pre-trained CNN as a feature extractor rather than training it from scratch provides the benefit of using it as a regularizer to guide the training and to reduce overfitting as well as that of transfer learning. The detailed architecture is discussed in the next section. The final part is the post-processing unit, which works

in accordance with the pre-processing one. If the task is to detect fully computer-generated images, the scores of the extracted patches are averaged. If the input is video, the scores of all frames are averaged. This average score is the final output.

User Interface

In this project, we have both users and an admin module. As soon as a user opens the website, the user is shown the login page and can register if the user is a new user. If the user is already registered, they can login with username and password.

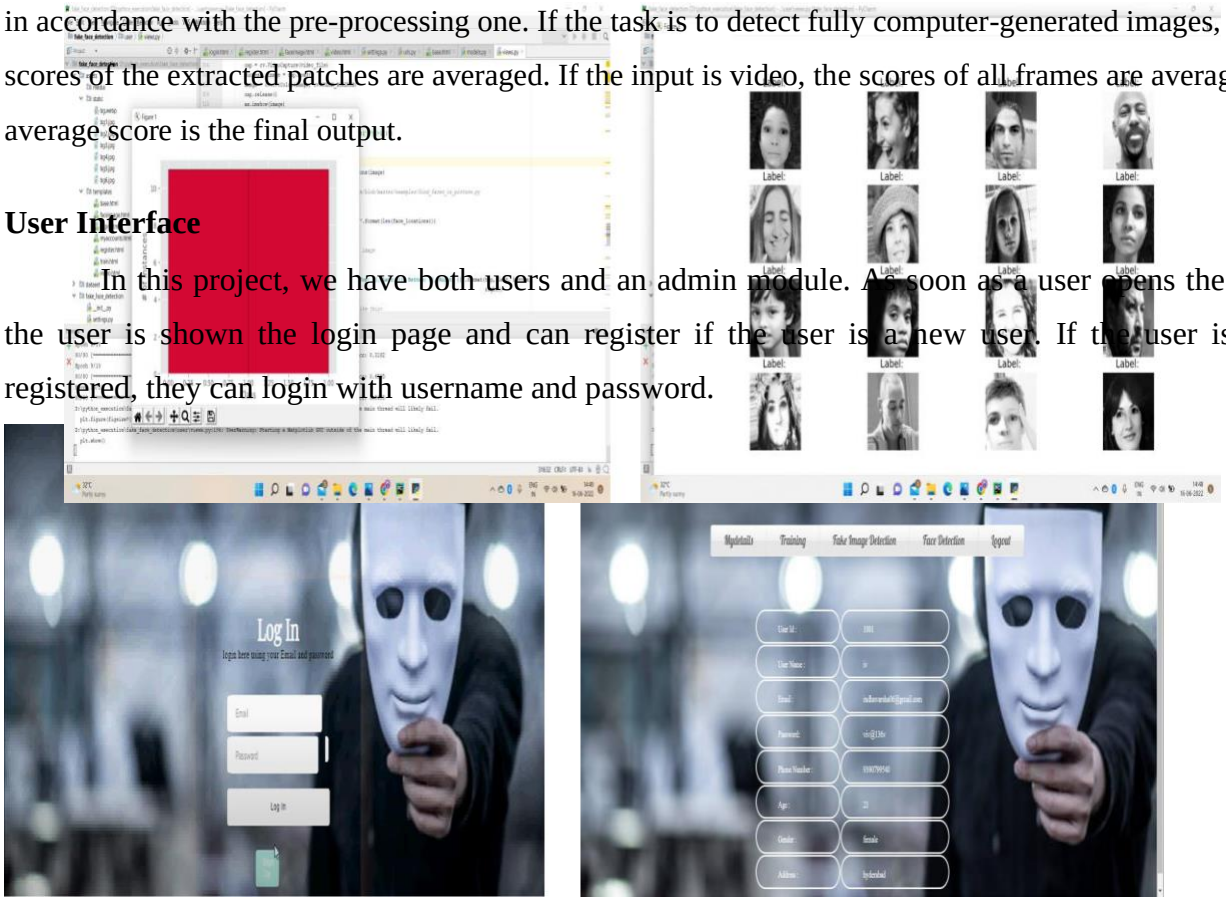


Figure 4.2: User Login and Registration

The user can register themselves by entering the details. Once the user logins into the account, he/she is shown the options like my details, training, fake image detection, face detection etc. The user can select an option; for example, the user can select the image and see is it fake or real.

Figure 4.3: Training for facial crop

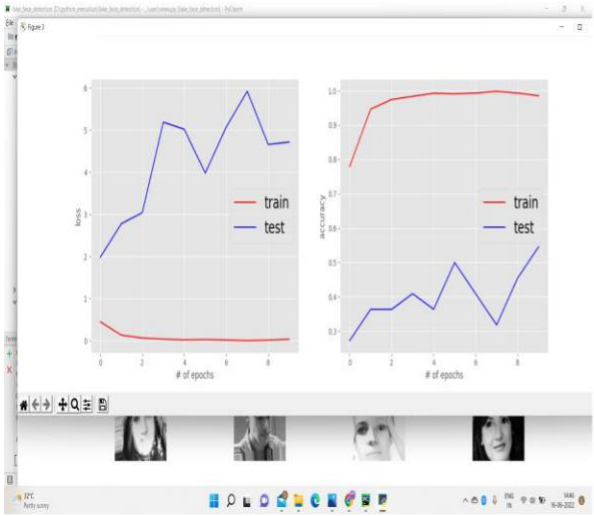


Figure 4.5: Graph of training data

Figure 4.4: Training data

Figure 4.6: Selecting image

The images in the database are displayed like this. Now users can click on the view prediction to know about the fake or real. This page will be opened. Now the user can see weather image is fake or real.

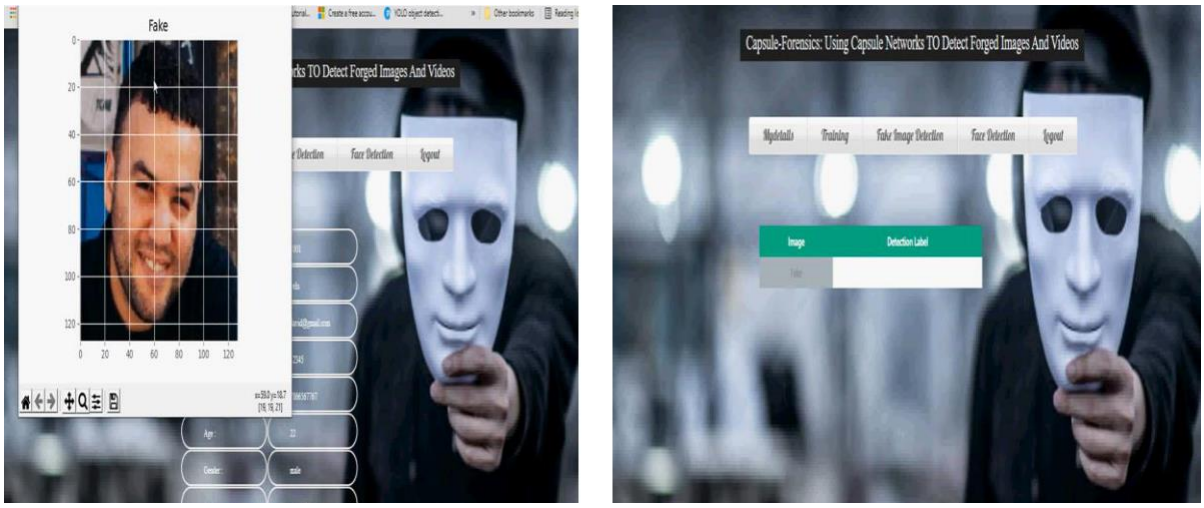


Figure 4.7: The selected image is pre-processed to crop facial area and Output

12

CONCLUSION AND FUTURE ENHANCEMENT

This paper developed an AVI embedded with a TensorFlow-based semi-supervised DL model to accurately auto-recognize an interviewee's true personality based on only 120 real samples of job applicants. Our APR approach achieved an accuracy above 90%, outperforming previous related laboratory studies whose accuracy ranged between 61% and 75% in the context of nonverbal communication. The high-performing APR used in this AVI can be adopted to supplement or replace self-reported personality assessment methods that can be distorted by job applicants due to the effects of social desire to be selected for employment. Previous related studies have found that multimodal features (image frames and audio) learned by deep neural networks can deliver better performances in predicting the big five traits than can unimodal features. In future work, we may combine our visual approach with prosodic features to learn how to recognize an interviewee's personality. Moreover, this study utilized a specific type of professional as participants, which may limit the generalizability of these experimental results. Future research should include a more diverse participant population.

REFERENCES

[1] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton, "Dynamic routing between capsules," in NIPS, 2017.

[2] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang, "Transforming auto-encoders," in

[3] Weize Quan, Kai Wang, Dong-Ming Yan, and Xiaopeng Zhang, "Distinguishing between natural and computer generated images using convolutional neural networks," IEEE TIFS, 2018.

[4] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner, "Face2Face: Real-time face capture and reenactment of RGB videos," in CVPR. IEEE, 2016.

[5] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Perez, Christian Richardt, Michael Zollh ´ ofer, and Chris- ¨ tian Theobalt, "Deep video portraits," in SIGGRAPH. ACM, 2018.

[6] Hadar Averbuch-Elor, Daniel Cohen-Or, Johannes Kopf, and Michael F Cohen, "Bringing portraits to life," ACM TOG, 2017.

[7] Joon Son Chung, Amir Jamaludin, and Andrew Zisserman, "You said that?," arXiv preprint arXiv:1705.02966, 2017.

[8] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman, "Synthesizing obama: learning lip sync from audio," ACM TOG, 2017.

[9] Ivana Chingovska, Andre Anjos, and Sebastien Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in BIOSIG, 2012.

[10] Tiago de Freitas Pereira, Andre Anjos, Jos ´ e Mario ´ De Martino, and Sebastien Marcel, "Can face anti- ´ spoofing countermeasures work in a real world scenario?," in ICB. IEEE, 2013.

[11] Yuezun Li, Ming-Ching Chang, Hany Farid, and Siwei Lyu, "In ictu oculi: Exposing AI generated fake face videos by detecting eye blinking," arXiv preprint arXiv:1806.02877, 2018.

[12] Jessica Fridrich and Jan Kodovsky, "Rich models for steganalysis of digital images," IEEE TIFS, 2012.

[13] Huy H Nguyen, Ngoc-Dung T Tieu, Hoang-Quoc Nguyen-Son, Vincent Nozick, Junichi Yamagishi, and Isao Echizen, "Modular convolutional neural network for discriminating between computer-generated images and photographic images," in ARES. ACM, 2018.