Face Recognition Using Deep Neural Network With Livenessnet

Nikhil kumar[1], Rithesh R[2], Ravishankara K[3]
[1, 2] Dept of CSE
[3]Professor, Dept of CSE
[1, 2, 3] Srinivas Institute of Technology, Mangalore

Abstract- *Continuous increase in the amount of population over the world leads to the increases the number of complex datasets over a period. This necessitates improving artificial intelligence algorithms for better and accurate categorization of data. Every person is unique in terms of face, although have the same structure such as noise, eyes, lips, etc. but it can vary strikingly. It's within this variance which lies in the distinguishing characteristics that can be used to identify one person from another. The "Face Recognition using DNN with LivenessNet" presents a face recognition method based on deep neural networks for liveness. Any algorithm is considered to be efficient only if it is robust and accurate. It provides accurate results with face spoofing quickly and efficiently. The main advantage of using this technique is identifying the uniqueness in the datasets by capturing the real-time face data through different modes & jitter. It provides accurate face recognition model which can be used for safety and security purpose.*

Keywords- Liveness Net, Deep Learning, Datasets, Convolutional Neural Networks.

I. INTRODUCTION

Due to some potential value for applications such as forensic science, automatic attendance, etc., face recognition is one of the most active research areas in the field of computer vision and machine learning. With the advancement of digital technologies, the demand for security to provide access control is increasing. Various face recognition algorithms have been proposed in the last few years, but due to high variability like pose, illumination, expression change, image resolution and other factors; face recognition has become too complex. The new solution incorporates biometrics that access people's physical features, such as iris, palm shapes, fingerprints, speech and face. Each one is having different characteristics that are distinguished in its own way based on the requirement. Therefore, this field is still active for research purpose.

In order to provide more good results, this product describes a face recognition approach based on deep neural networks to evaluate human face detection with vividness. Next, the identification of the face should be viewed as a special object-class identification event. It looks for the position and dimensions of all the features belonging to a class called a face, such as eyes, the distance between them, etc.

II. LITERATURESURVEY

In the research done by Krishnapriya, report on a methodical investigation into differences in face recognition accuracy between African-American and Caucasian image cohorts of the MORPH dataset. They found that, for all four matchers considered, the impostor and the genuine distributions are statistically significantly different between cohorts. For a fixed decision threshold, the African-American image cohort has a higher false match rate and a lower false non-match rate. ROC curves compare verification rates at the same false match rate, but the different cohorts achieve the same false match rate at different thresholds. This means that ROC comparisons are not relevant to operational scenarios that use a fixed decision threshold. They showed that, for the ResNet matcher, the two cohorts have approximately equal separation of impostor and genuine distributions. Using ICAO compliance as a standard of image quality,they found that the initial image cohorts have unequal rates of good quality images. The ICAO-compliant subsets of the original image cohorts show improved accuracy, with the main effect being to reducing the low-similarity tail of the genuine distributions [1].

In the research done by Huang for face analysis often exhibit highly-skewed class distribution, i.e., most data belong to a few majority classes, while the minority classes only contain a scarce amount of instances. To mitigate this issue, contemporary deep learning methods typically follow classic strategies such as class re-sampling or cost-sensitive training. In this paper, we conduct extensive and systematic experiments to validate the effectiveness of these classic schemes for representation learning on class-imbalanced data. We further demonstrate that more discriminative deep representation can be learned by enforcing a deep network to maintain inter-cluster margins both within and between

classes. This tight constraint effectively reduces the class imbalance inherent in the local data neighborhood, thus carving much more balanced class boundaries locally. We show that it is easy to deploy angular margins between the cluster distributions on a hypersphere manifold. Such learned Cluster-based Large Margin Local Embedding (CLMLE), when combined with a simple k-nearest cluster algorithm, shows significant improvements in accuracy over existing methods on both face recognition and face attribute prediction tasks that exhibit imbalanced class distribution [2].

In the research done by Lior shkiller and Galit yovel, Face recognition is a computationally challenging task that humans perform effortlessly. This remarkable ability was better for familiar faces than unfamiliar faces. To account for humans superior ability to recognize familiar faces, current theories suggest that different features are used for the representation of familiar and unfamiliar faces. In the current study, they applied a reverse engineering approach to reveal which facial features are critical for familiar face recognition. In contrast to current views, they have discovered that the same subset of features that are used for matching unfamiliar faces, are also used for matching as well as recognition of familiar faces. Also show that these features are also used by a deep neural network face recognition algorithm. So they proposed a new framework that assumes similar perceptual representation for all faces and integrates cognition and perception to account for humans' superior recognition of familiar faces [3].

In the search done by Masi and Lacopo,identify two issues as key to developing effective face recognition systems: maximizing the appearance variations of training images and minimizing appearance variations in test images. The former is required to train the system for whatever appearance variations it will ultimately encounter and is often addressed by collecting massive training sets with millions of face images. The latter involves various forms of appearance normalization for removing distracting nuisance factors at test time and making test faces easier to compare. Describe novel, efficient face-specific data augmentation techniques and show them to be ideally suited for both purposes. By using knowledge of faces, their 3D shapes, and appearances, we show the following: (a) artificially enrich training data for face recognition with face-specific appearance variations. (b) This synthetic training data can be efficiently produced online, thereby reducing the massive storage requirements of large-scale training sets and simplifying training for many appearance variations Together, with additional technical novelties, we describe a highly effective face recognition pipeline which, at the time of submission, obtains state-of-the-art results across multiple benchmarks [4].

In the research done by Manik Sharma and J Anuradha, they have observed that Facebook has developed an uncanny ability to recognize people in photographs. They had to tag people in photos by clicking on them and typing their name. Now as soon as we upload a photo, Facebook tags everyone on its own. Facebook can recognize faces with 98% accuracy which is pretty much as good as humans can do. This technology is called Face Detection. Face detection was a popular topic in biometrics. They have surveillance cameras in public places for video capture as well as security purposes. The main advantages of this algorithm over other are uniqueness and approval. They need speed and accuracy to identify. But face detection is really a series of several related problems: First, look at a picture and find all the faces in it. Second, focus on each face and understand that even if a face is turned in a weird direction or in bad lighting, it was still the same person. Third select features which can be used to identify each face uniquely like size of the eyes, face etc. Finally, compare these features to data they have to find the person name. As a human, your brain is wired to do all of this automatically and instantly. In fact, humans are too good at recognizing faces. Computers are not capable of this kind of high-level generalization this process separately. The growth of face detection is largely driven by growing applications such as credit card verification, surveillance video images, authentication for banking and security system access [5].

In the research done by Deng and Jiankang, face representation using Deep Convolutional Neural Network (DCNN) embedding is the method of choice for face recognition. Current state-of-the-art face recognition systems can achieve high accuracy on existing in-the-wild datasets. However, most of these datasets employ quite limited comparisons during the evaluation, which does not simulate a real-world scenario, where extensive comparisons are encountered by a face recognition system. They propose two large-scale datasets and define an extensive comparison metric for an unbiased evaluation of deep face recognition models. To ensure fair comparison during the competition they define light-model track and large-model track, respectively. Each track has strict constraints on computational complexity and model size. To the best of our knowledge, this is the most comprehensive and unbiased benchmarks for deep face recognition [6].

III. OBJECTIVE

The aim of our project is to show liveness detection, including what it is andwhy it is need to improve face recognition systems. Then it will be reviewed the dataset and will used to perform liveness detection, including how to build to a dataset for liveness detection and our example real versus fake face images. Here also reviewed project structure for the

liveness detector project as well. To create the liveness detector, deep neural network will be trained which is capable of distinguishing between real versus fake faces.

IV. IMPLEMENTATION

Step #1: Gather Your Dataset

The first component of building a deep learning network is to gather our initial dataset. We need the images themselves as well as the labels associated with each image. These labels should come from a finite set of categories, such as: categories = dog, cat, panda. Furthermore, the number of images for each category should be approximately uniform (i.e., the same number of examples per category). If we have twice the number of cat images than dog images, and five times the number of panda images than cat images, then our classifier will become naturally biased to overfitting into these heavily-represented categories. Class imbalance is a common problem in machine learning and there exist a number of ways to overcome it. We'll discuss some of these methods later, but keep in mind the best method to avoid learning problems due to class imbalance is to simply avoid class imbalance entirely.

Step #2: Split Your Dataset

Now that we have our initial dataset, we need to split it into two parts:

1. A training set
2. A testing set

A training set is used by our classifier to "learn" what each category looks like by making predictions on the input data and then correct itself when predictions are wrong. After the classifier has been trained, we can evaluate the performing on a testing set.It's extremely important that the training set and testing set are independent of each other and do not overlap! If you use your testing set as part of your training data, then your classifier has an unfair advantage since it has already seen the testing examples before and "learned" from them. Instead, you must keep this testing set entirely separate from your training process and use it only to evaluate your network. Common split sizes for training and testing sets include 66:6%33:3%, 75%=25%, and 90%=10%These data splits make sense, but what if you have parameters to tune? Neural networks have a number of knobs and levers (ex., learning rate, decay, regularization, etc.) that need to be tuned and dialed to obtain optimal performance. We'll call these types of parameters hyper parameters, and it's critical that they get set properly. In practice, we need to test a bunch of these hyper parameters and identify the set of parameters that works the best. You might be tempted to use your testing data to tweak these values, but again, this is a major no-no! The test set is only used in evaluating the performance of your network. Instead, you should create a third data split called the validation set. This set of the data (normally) comes from the training data and is used as "fake test data" so we can tune our hyper parameters. Only after have we determined the hyper parameter values using the validation set do we move on to collecting final accuracy results in the testing data. We normally allocate roughly 10-20% of the training data for validation. If splitting your data into chunks sounds complicated, it's actually not.

Step #3: Train Your Network

Given our training set of images, we can now train our network. The goal here is for our network to learn how to recognize each of the categories in our labeled data. When the model makes a mistake, it learns from this mistake and improves itself. So, how does the actual "learning" work? In general, we apply a form of gradient descent.

Step #4: Evaluate

Last, we need to evaluate our trained network. For each of the images in our testing set, we present them to the network and ask it to predict what it thinks the label of the image is. We then tabulate the predictions of the model for an image in the testing set.

Finally, these model predictions are compared to the ground-truth labels from our testing set. The ground-truth labels represent what the image category actually is. From there, we can compute the number of predictions our classifier got correct and compute aggregate reports such as precision, recall, and f-measure, which are used to quantify the performance of our network as a whole.

Detection of real and fake images:

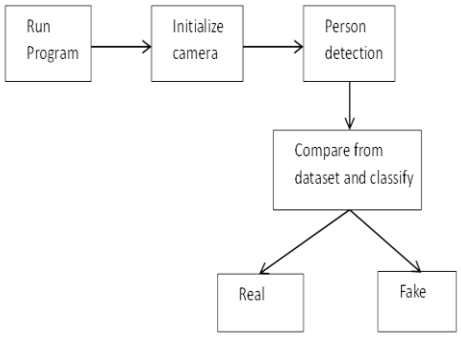

Figure 1: Detection of real and fake images.

ALGORITHM

Convolutional Neural Networks (CNN)

Layer types:

There are many types of layers used to build Convolutional Neural Networks, but the ones you are most likely to encounter include:

- Convolutional (CONV)
- Activation (ACT or RELU, where we use the same of the actual activation function)
- Pooling (POOL
- Fully-connected (FC)
- Batch normalization (BN)
- Dropout (DO)

The last layer in a CNN uses these higher-level features to make predictions regarding the contents of the image. In terms of deep learning, an (image) convolution is an element-wise multiplication of two matrices followed by a sum.

1. Take two matrices (which both have the same dimensions).
2. Multiply them, element-by-element (i.e., not the dot product, just a simple multiplication).
3. Sum the elements together.

V. ANALYSIS

Table 1**:** Result Analysis Table

Person name	No. of Testing augmentations	Result accuracy for fake	Result Accuracy for real
Person1	100	0.98	0.94
Person2	100	1	0.96
Person3	100	0.95	0.93
Person4	100	0.98	0.92
Person5	100	0.96	0.95
Person6	100	1	0.92
Person7	100	0.97	0.94
Person8	100	0.97	0.95
Person9	100	1	0.94
Person10	100	1	0.96

Average overall accuracy for real=0.9410=94.10%
Average overall accuracy for fake=0.9810=98.10%

VI. CONCLUSION

The proposed method for detecting faces can recognize faces with excellent accuracy and resilience. The begining stage in developing the suggested liveness detector facial recognition model is to gather data for training the LivenessNet model by holding a smartphone in front of the camera. The dataset is derived from footage captured with the camera directly. The dataset is identical to the genuine dataset for recognition. The liveness of the frame is initially recognized using the LivenessNet model after the dataset has been prepared. The deep learning model built on the training dataset is then used to recognize the discovered real face.

REFERENCES

[1] Krishnapriya, K. S., et al. "Characterizing the variability in face recognition accuracy relative to race." preprint arXiv: 1904.07325 (2019).
[2] Huang, Chen, et al. "Deep imbalanced learning for face attribute prediction," IEEE transactions on pattern analysis and machine intelligence (2019).
[3] Abudarham, Naphtali, LiorShkiller, and GalitYovel. "Critical features for face recognition." Cognition 182 (2019): 73-83
[4] Masi, Iacopo, et al. "Face-specific data augmentation for unconstrained face recognition." International Journal of Computer Vision 127.6-7 (2019): 642-667.
[5] Manik Sharma, J Anuradha, H KManne and G S Kashyap (2017), “Facial detection using deep learning”, School of Computing Science and Engineering, VIT University, Vellore - 632014, India (DOI: 10.1088/1757-899X/263/4/042092.
[6] Deng, Jiankang, et al. "Lightweight face recognition challenge," Proceedings of the IEEE International Conference on Computer Vision Workshops (2019).