Convolutional Neural Network and application in Face Liveness Detection using Python and OpenCV

Gia Hy Nguyen - 101922778
Faculty of Science, Engineering and Technology
Swinburne University of Technology
Hawthorn, Victoria 3122

***Abstract* – Liveness detection, which is also known as Face Anti-Spoofing, is a task of distinguishing fake and real faces in face recognition systems. This report discusses a method to implement Liveness Detection by using a deep learning algorithm called Convolutional Neural Network. A practical experiment is also conducted to visualise the output of the application, with the help of Python and OpenCV library.**

***Keywords* – artificial neural network, face spoofing, liveness detection, machine learning, deep learning.**

I. INTRODUCTION

Since the outbreak of AI and Machine Learning in recent years, many face recognition systems were built to serve for multiple purposes such as automatic timekeeping/attendance system, phones unlocking, home security, banking, … and many other useful applications. With the rising popularity of face recognition, there is also an increase in the number facial spoofing techniques to break into the systems that use face recognition technology. To prevent the attacks, Face Liveness Detection has been developed to protect the users from loosing their assets (with home security, banking systems, …) or any other unintended consequences.

Face Liveness Detection is an ability to prevent face spoofing techniques. It can distinguish and detect if the face in front of the camera is real or fake. There are several common liveness detection approaches such as Texture Analysis, 3D Face Shape, Optical Flow Algorithm, … but the approach that will be demonstrated in the paper is Convolutional Neural Network. In this approach, we treat Face Liveness Detection as a binary classification problem. A convolutional neural network will be constructed and trained to distinguish real/fake faces from a real-time video surveillance system.

II. CONVOLUTION NEURAL NETWORK

CNN as briefly explained earlier, is a deep learning algorithm to differentiate images. The common features of input images can be learnt and extracted according to the depth and combination of layers in the CNN.

To explore the definition of convolutional neural network and understand how it can be applied in Liveness Detection, we can start with its basic components. Basically, a CNN contains convolution layer, pooling layer and fully connected layer.

A. Convolution Layer

This layer is named after a mathematical linear operation between matrixes called convolution. Convolution Layer uses a kernel with custom dimension to slide on the input image and apply matrix multiplication operation between the kernel and the region

of the input image that is covered by the kernel's dimension. The kernel slides from left to right, top to bottom until it traverses over the whole input image. A convolved feature is the output of this layer.

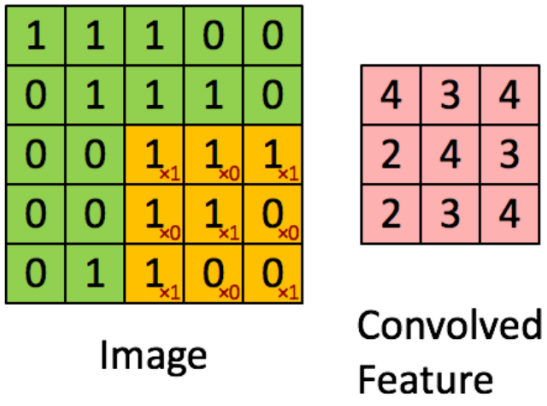

Figure 1. A 5x5x1 image is convolved with a 3x3x1 kernel, and the output is a 3x3 matrix

In case the input images have multiple channels (e.g. RGB images), the kernel will have the same depth with the input images. Convolution is performed on the same level of channels between the kernel and the input images, then the results will be summed with the bias to form the final one-depth channel convolved feature output, which will be passed into deeper convolution layers and the same process will happens again.

The final convolved feature output contains the high-level features (e.g. edges, curves, lines, …) from the input images, provide us the understanding and analysis of the dataset, therefore we can extract the similarities and differences between them.

B. Pooling Layer

The role of Pooling Layer is to decrease the size of the final convolved feature output, hence minimize the required computations and parameters when processing the data. Additionally, Pooling Layer can also generate the summaries of high-level features that are extracted from Convolution Layer, which helps to reinforce the model because later operations will be applied on the summarised features, not the exact positioned features, hence avoid the variations in analysing the position of the features.

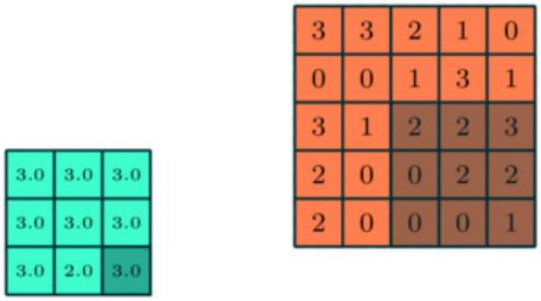

Figure 2. A 3x3 pooling layer applied on a 5x5 convolved feature output

The most two common types of Pooling Layer are Max Pooling and Average Pooling. In Max Pooling, the maximum value in the region of the convolved feature output covered by the pooling layer will be extracted. While in Average Pooling, the average of all the values covered in the region of pooling layer will be selected. Max Pooling was proved to be better than Average Pooling in other CNN related papers.

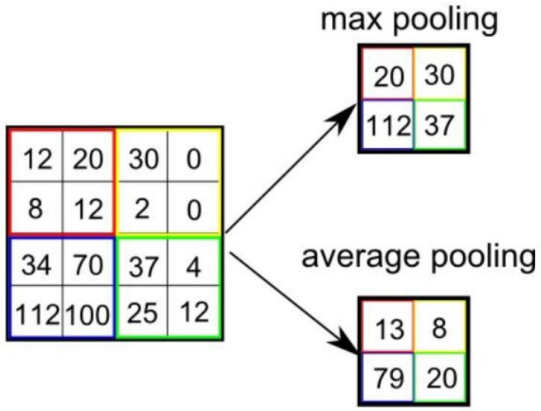

Figure 3. Max Pooling and Average Pooling

After going through the two layers above, the main features of the input images are extracted and understood by the model. The next step is to process the output and feed it into the last phase: Fully Connected Layer.

C. *Fully Connected Layer (classification layer)*

With the help from Convolution Layer and Pooling Layer, we have the input images converted into a suitable form that can be fed into a Fully Connected Layer, or also known as Multi-level Perceptron.

This layer is a very basic layer in every artificial neural network, it is used for learning the non-linear combinations of high-level features and produces the final classification result.

However, Fully Connected Layer only works with flatten input vector, hence we have to unroll the output generated by Pooling Layer before feeding it into a feed-forward neural network. Activation function of each node in the layer can be selected differently according to the task that we are doing. The common activation functions for fully connected layer are Sigmoid, ReLU, ELUs, Softmax, …

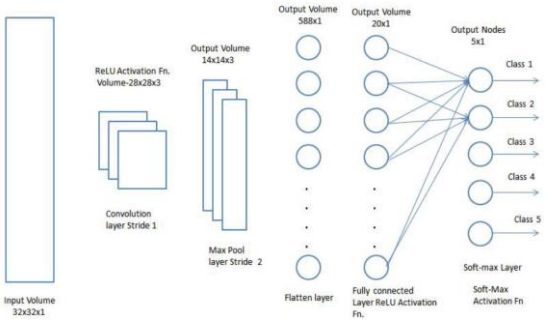

Figure 4. Fully Connected Layer

III. APPLICATION IN LIVENESS DETECTION

To visualise the power of Convolutional Neural Network, I have conducted an experiment which uses CNN to resolve the face spoofing problem. As we have discovered above, CNN has a very special potential in processing and mapping a 2D image to an output variable. For any machine learning problem that takes images as input, CNN is first thing that developers should think about. One of the biggest benefits of using CNN is that it can detects the important features of an image with a very small number of parameters compared to the predecessors, hence reduce the computational cost and can be applied on real-time devices.

Since Liveness Detection works with images and it requires a lightweight solution so that it can run on real-time video surveillance systems, CNN turns out to be one of the best candidates. In my experiment, I treat the liveness detection task as a binary classification problem. A comprehensive structure of CNN will be built and trained to be capable of distinguishing real and fake faces.

A. *Module 1: Build "LivenessNet" – the CNN for liveness detection*

The network that I built for this experiment contains 4 convolution layers, 2 max pooling layers, and 2 dense layers. The structure and input/output dimension of each block are represented in the figure below:

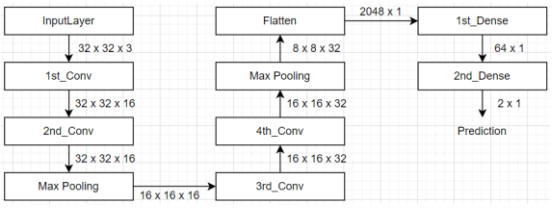

Figure 5. LivenessNet structure

The activation function of all blocks is ReLU, except the final Dense layer, which uses Softmax. After the 4 convolution layers and 2 max pooling layers, the model is expected to be capable of extracting and understanding the high-level features of input images, distinguishing the similarities and differences between the data. Finally, with the help of the last 2 Dense layers, the model will be able to classify the input images into 2 separate

classes, then answers the question of determining real and fake faces.

B. *Module 2: Train "LivenessNet"*

After having the dataset and LivenessNet structure prepared, I started to train the network. The dataset that I used (which will be discussed in the next section) contains 227 images in total, and 75% of them will be extracted for training set, and the remaining 25% is allocated for validation set.

Images are resized to 32x32 dimension before being fed into the network. Augmentation is performed to enrich the dataset and avoid overfitting.

The optimizer used for training is Adam, with the learning rate of 10^{-5}, batch size equals 8 and number of training epochs is 50.

IV. DATASET

There are many different types face spoofing attack such as photograph/video, 3D mask, high-resolution prints, … but within the context of this paper, I only focus on distinguishing spoofed faces on a mobile screen.

The dataset in this paper is constructed by recording my real face and my spoofed face in two different videos, and then the frames will be extracted and applied face detection to create the dataset for this experiment. Below is the statistics of the dataset:

Class	Number of images
Fake	112
Real	115
	Total: 227

Figure 6. Dataset

Since this is a personal experiment without research budget, the data is slightly biased towards my pictures. Ideally, faces with different ethnicities and angles should be captured to improve the model's performance.

V. EXPERIMENT RESULTS

After 50 training epochs, I achieved 94% accuracy on the training set and 100% for the validation set. The training/validation loss and accuracy over each epoch is captured in the figure below:

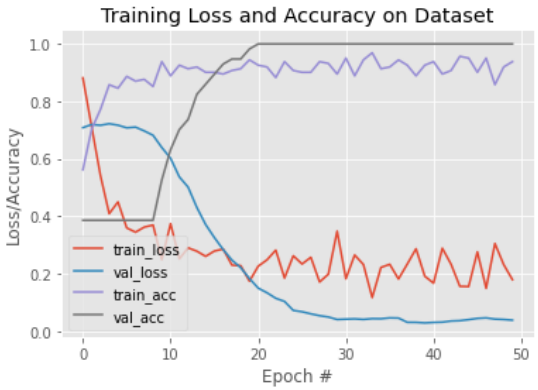

Figure 7. Loss and accuracy plot

The precision and recall of this model are also very impressive, they both achieve 1.0. Hence, the F1-score of the model is also 1.0. Below is the report of model's performance from the perspective of confusion matrix evaluation:

	Precision	Recall	F1-score
Fake	1.00	1.00	1.00
Real	1.00	1.00	1.00
Accuracy			1.00
Macro avg	1.00	1.00	1.00
Weighted avg	1.00	1.00	1.00

To visualise the output of the experiment, I have written a script to run the model on my laptop's camera and perform the real-time liveness detection on it. This is the closest simulation of the usage of Liveness Detection when applying it in automatic timekeeping and attendance system. The demonstration is recorded and can be accessed via here. Below is the screenshot of the application running:

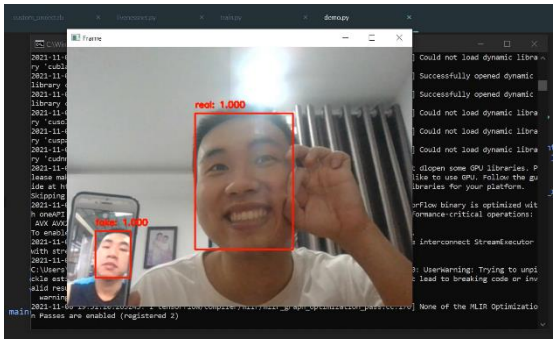

Figure 8. Application demonstration

VI. CONCLUSION

Convolutional Neural Network can be very powerful in any machine learning tasks that take images as an input source. The reason behind this advantage is because CNN reduces a large number of parameters compared to a normal feed-forward network, due to the mechanism of weight sharing between convolution layer and pooling layer. CNN can capture and understand well the high-level features of input images at a level of human brain. It acts as a brilliant feature extractor.

Via the research and the conducted experiment, we have verified the power and the usages of Convolutional Neural Network in real-life problems. The model was able to achieve a very impressive result with the absolute accuracy. CNN is also a very lightweight structure that can be embedded and applied on real-time surveillance systems.

VII. ACKNOWLEDGEMENT

I would like to thank Adrian Rosebrock for providing numerous useful blogs and courses online about Machine Learning, Deep Learning. Secondly, I would also like to express my appreciation for Prof. Andrew Ng and Tiep V. Huu for their extraordinary sharing about Convolutional Neural Network. Lastly, thanks to Google Colab for offering free GPU usage so that I can train the model for this paper.

VIII. REFERENCES

[1] Adrian Rosebrock. (2019). *Liveness Detection with OpenCV*, PyImageSearch. Retrieved from https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/

[2] Albawi, S & Mohammed, T. (2017). *Understanding of a Convolutional Neural Network*. Retrieved from https://www.researchgate.net/publication/319253577_Understanding_of_a_Convolutional_Neural_Network

[3] Electronic Identification. (2021). *Face Liveness Detection for spoofing face recognition*, Electronic Identification. Retrieved from https://www.electronicid.eu/en/blog/post/face-liveness-detection-spoofing-face-recognition/en

[4] Sumit Saha. (2018). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, Towards Data Science. Retrieved from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[5] Thales Group. (2020). *Liveness in biometrics: spoofing attacks and detection*, Thales Group. Retrieved from https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/inspired/liveness-detection