electronics

MDPI

Article

Design and Implementation of Deep Learning Based Contactless Authentication System Using Hand Gestures

Aveen Dayal [1], Naveen Paluru [2], Linga Reddy Cenkeramaddi [1,*], Soumya J. [3] and Phaneendra K. Yalavarthy [2]

[1] Department of Information and Communication Technology, University of Agder, 4879 Grimstad, Norway; aveendayal97@gmail.com
[2] Department of Computational and Data Sciences, Indian Institute of Science, Bangalore 560012, India; naveenp@iisc.ac.in (N.P.); yalavarthy@iisc.ac.in (P.K.Y.)
[3] Birla Institute of Technology and Science-Pilani, Hyderabad 500078, India; soumyaj@hyderabad.bits-pilani.ac.in
* Correspondence: linga.cenkeramaddi@uia.no

Abstract: Hand gestures based sign language digits have several contactless applications. Applications include communication for impaired people, such as elderly and disabled people, health-care applications, automotive user interfaces, and security and surveillance. This work presents the design and implementation of a complete end-to-end deep learning based edge computing system that can verify a user contactlessly using 'authentication code'. The 'authentication code' is an 'n' digit numeric code and the digits are hand gestures of sign language digits. We propose a memory-efficient deep learning model to classify the hand gestures of the sign language digits. The proposed deep learning model is based on the bottleneck module which is inspired by the deep residual networks. The model achieves classification accuracy of 99.1% on the publicly available sign language digits dataset. The model is deployed on a Raspberry pi 4 Model B edge computing system to serve as an edge device for user verification. The edge computing system consists of two steps, it first takes input from the camera attached to it in real-time and stores it in the buffer. In the second step, the model classifies the digit with the inference rate of 280 ms, by taking the first image in the buffer as input.

Keywords: hand gestures recognition; security; edge computing; deep learning; neural networks; contactless authentication; camera based authentication

Citation: Dayal, A.; Paluru, N.; Cenkeramaddi, L.R.; J., S.; Yalavarthy, P.K. Design and Implementation of Deep Learning Based Contactless Authentication System Using Hand Gestures. *Electronics* **2021**, *10*, 182. https://doi.org/10.3390/electronics10020182

Received: 2 December 2020
Accepted: 12 January 2021
Published: 15 January 2021

1. Introduction

Contactless biometric authentication has become a necessity at this hour as it is perceived to be not only more hygienic but secure and efficient. Biometrics can be divided into two classes, physiological biometrics and behavioral biometrics [1]. Physiological biometrics usually includes fingerprints [2], facial features [3], palm prints [4], retinas [5], ears [6], and irises [7]. Behavioural biometrics usually consists of keystrokes [8], signatures [9], and gaits [10]. Voice biometric can be classified as both because it includes features belonging to both the classes [1].

Before the era of Deep Learning, biometric authentication was mainly based on hand-crafted features that were extracted using methods such as scale invariant feature transform (SIFT) [11], wavelet [12], etc. With the advent of deep learning in this decade, the biometric authentication field is completely transformed. Most contemporary biometric authentication systems use convolution neural networks and different variants of them. A convolutional neural network (CNN) is a type of deep multilayer artificial neural network. They are widely used in all computer vision tasks because of their convolution process that obtains an effective representation of the input images directly from raw pixels with little to none preprocessing and can easily recognize visual patterns [13]. The representations learned by the CNN models are that of visual features and are effective when compared to the handcrafted features [14]. These networks are being applied in a variety of applications,

such as object detection tasks, speech recognition tasks, unmanned aerial vehicles (UAVs), and unmanned ground vehicles (UGVs) [15]. Below mentioned are some of the contemporary biometric authentication systems that use deep learning models, ref. [16] uses CNN for facial biometric authentication, ref. [17] authenticates a user, based on fingerprints using CNN, ref. [18] uses graph neural networks and CNN for palmprint recognition, ref. [19] uses deep neural networks to identify and authenticate a user based on voice, ref. [20] uses deep learning to recognize gaits, ref. [21] recognizes signature biometric using deep learning, etc. Even though deep learning has brought a lot of progress in the field of biometric authentication there are still several challenges to overcome. Some of the challenges are, more challenging datasets have to be developed to train the models, the need for interpretable deep learning models, real-time deployment of the models, memory-efficient models, security and privacy issues, etc. [22].

Also in the security space, the most dominant usage of CNNs had been in the intrusion detection [23]. These networks primarily use object detection modules and the recent development of CNNs has been proven to be effective for object detection [24]. Many of these CNNs are memory hungry due to a high number of parameters (hundreds of millions) and have high computational complexity. This impedes the deployment in the edge computing device, which is a typical requirement in edge computing/cyber-physical system (CPS) [25]. So, the development of memory-efficient CNNs has taken the forefront in recent years [25], which can provide a compact model that is deployable in edge computing/CPS arenas and have hardware-level support that is needed.

Thus to tackle the real-time deployment and memory efficiency challenges of the deep learning models, we propose an end-to-end contactless authentication system that verifies a user by validating the 'authentication code' using a memory-efficient CNN model. The 'authentication code' which is unique for each user and is an 'n' digit numeric code with each digit having a range of 0–9. The task of automatically classifying an input image into one of the given classes using convolution neural networks is known as image classification task [26]. There are quite a few standard image classification datasets namely 'ImageNet Large Scale Visual Recognition Challenge' [27], 'CIFAR 10', 'CIFAR 100' [28], 'Oxford IIIT Pet' [29], 'Oxford Flowers' [30], 'MNIST' [31], etc. There are also quite a few deep learning models available today that achieve high performance on these image classification datasets using different variants of convolution neural networks e.g., the Sharpness-Aware minimization procedure on CNN model achieves 88.61% top-1 accuracy on ILSVRC-2012 dataset [32], the EnAET model with an error rate of 1.99% on CIFAR-10 dataset [33], the Big Transfer (BiT) model with an accuracy of 99.4% on CIFAR-10 dataset and 87.5% top-1 accuracy on ILSVRC-2012 dataset [34], the Branching and Merging convolution network with homogeneous filter capsules model with an accuracy of 99.84% on MNIST dataset [35]. These models consist of several convolution layers, max-pooling layers, and different regularization layers like drop out, and L2 regularization. But all the aforementioned models are very large models that require more memory and greater model inference time. Memory efficient CNN's execution without having any compromise on the accuracy has been a challenge, especially when the inference has to be performed on an edge computing device/CPS. The state-of-the-art performance has been achieved with the help of complex models that require more computational resources [25]. In this work, we develop a memory-efficient CNN model that suits best for edge computing devices. We also compare the performance of the proposed memory efficient CNN model with the MobileNetV2 model, which is the state of the art model for memory-efficient CNN, and show that the proposed network outperforms the existing MobileNetV2 model.

2. Materials and Methods

2.1. Dataset

We have utilized 'Sign Language Digits Dataset' [36] for the training of the proposed CNN and MobilenetV2. The dataset consists of a total of 2062 samples. There are 10 classes from digit 0 to digit 9. Samples from each class are shown in Figure 1. Each sample is of the

size [100 × 100] pixels. Table 1 shows dataset details with the total sample count per class. The entire dataset is split into three parts i.e. training data, validation data, and test data. Since there are 10 classes, the test data is split in such a way that there are equal numbers of samples in each class. We randomly selected 41 samples from each class leading to a total of 410 samples for the test set, which is 19.88% of the entire dataset. The rest of the dataset is split into training and validation sets with 20.07% and 60.03% samples of the dataset respectively.

Figure 1. Sample images of the dataset [36]. The number below each image represents the decoded sign.

Table 1. Dataset [36] details. Example Images in the dataset are given in Figure 2.

Class	Number of Samples	Number of Training	Number of Validation	Number of Testing
0	205	123	41	41
1	206	124	41	41
2	206	124	41	41
3	206	124	41	41
4	207	124	42	41
5	207	124	42	41
6	207	124	42	41
7	206	124	41	41
8	208	125	42	41
9	204	122	41	41
Total	2062	1238	414	410

As mentioned above, the image size of each sample in the dataset is [100 × 100] pixels. These image samples are resized (upscaled) to [256 × 256] pixel size using 'Bicubic Interpolation' technique [37]. Here we upscale the images to [256 × 256] pixel size because of the real-time testing constraints as explained in Section 2.4. The resized image samples are then used as input images to train the deep learning model.

2.2. Proposed End to End System for Contactless Authentication

In this work we use CNNs for providing contactless authentication code from input images of hand gestures of sign language digits. The whole system is composed of two steps namely capture of images and classification (digit recognition) of the captured images. The entire classification task used in the system is shown in Figure 2. As shown in Figure 2 the input image to the system is first resized to [256 × 256] pixel size and is then fed into the deep learning model for digit recognition.

Figure 2. The classification (digit recognition) task used in the proposed end to end system for contactless authentication. The architecture details of the deep learning model are shown in Figure 4 and in Table 5.

2.3. Lightweight Deep Learning Models for Hand Gestures Recognition

This section discusses the lightweight deep learning based CNN's suitable for edge computing devices for hand gesture recognition. Both the proposed model based on the bottleneck module which is inspired by the deep residual networks, and MobilenetV2 (for comparison) are discussed.

2.3.1. Mobilenetv2

MOBILE-NET-V2 architecture is the state-of-the-art lightweight CNN that is superior compared to other models and shown to provide improved performance especially in tasks like object recognition [38]. The network has depth-wise separable convolution layers (as shown in Figure 3), which act as efficient building blocks. The hypothesis of the network was to achieve an efficient encoding between intermediate input and output through these bottlenecks.

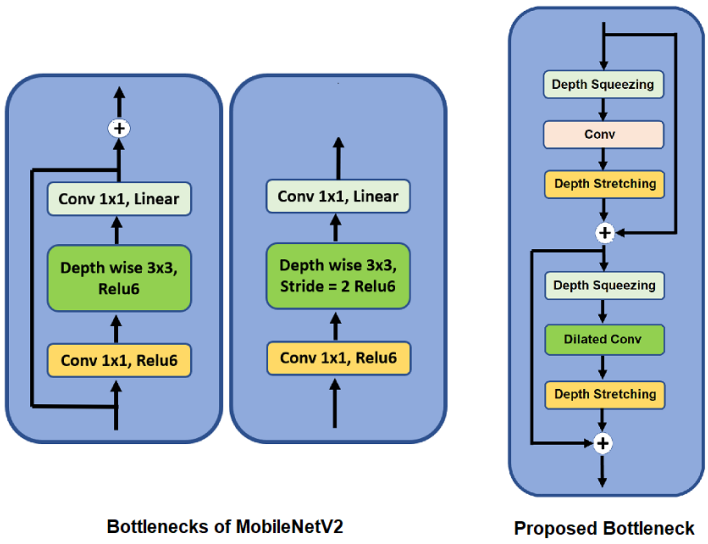


Figure 3. Comparison between the proposed bottleneck and the bottleneck of MobileNetV2. For convolution with stride 2, the bottleneck of MobileNetV2 does not have the residual connection. The details of the proposed bottleneck are given in Section 2.3.2.

We incrementally train the MobileNetV2 model using two different methods of transfer learning namely feature extraction and fine-tuning. We first train the model with the feature extraction method and after that, we continue training the model with the fine-tuning method. Both of these methods are briefly described below:

1. Feature extraction: In this method, we use the MobileNetV2 model pre-trained on the ImageNet dataset to work as a feature extractor. We do not include the final dense layer, 'classification layer' of the MobileNetV2 model because the number of classes for the ImageNet dataset is different from the 'Sign Language Digit Dataset' we use. So in total, we use 154 pre-trained layers of the MobileNetV2 model as the

base model. The output from this base model is the learned features which is a 4 dimensional tensor of size [None, 8, 8, 1280]. We then flatten the output of the base model into a 2-dimensional matrix of size [None, 1280] using 'Global Average Pooling 2D function' [39]. After this layer we add a Dense Layer which is the classification layer for the dataset we use. Therefore using the Feature extraction method we do not train the base model (MobileNetV2, excluding its final layer) instead, we use the base model to extract features from the input sample and then use these extracted features as input to the dense layer (the added classification layer according to the 'Sign Language Digit Dataset') and just train the dense layer on our dataset. The following hyper-parameter values are used to train the MobileNetV2 model using this method, 'RMS Prop optimizer' [40] with a learning rate of 0.0001, batch size of 32 for a total of 10 epochs. The architecture and the total number of parameters (model weights) used for training the MobileNetV2 model using this method are shown in Tables 2 and 3 respectively.

Table 2. The architecture used to train MobileNetV2 model using feature extraction and fine-tuning methods.

Layer Type	Output Size
MobileNetV2 (Base Model)	(None, 8, 8, 1280)
Global_Average_Pooling	(None, 1280)
Output	(None, 10)

Table 3. Comparison of transfer learning methods on MobileNetV2 model utilized in this work Section 2.3.1 in terms of parameters.

Method	Total no. of Parameters	Trainable	Non-Trainable
Feature extraction method	2,270,794	12,810	2,257,984
Fine-tuning method	2,270,794	2,236,682	34,112

2. Fine-tuning: In this method, we use the same model architecture as the above method, the MobileNetV2 model (excluding its classification layer) pre-trained on the ImageNet dataset is the base model and an added dense layer (classification layer according to our dataset). The only difference is that in this method, we train a few layers of the base model along with the final added dense layer on the 'Sign Language Digit Dataset'. So in total, we freeze the first '99' layers and start training from the layer '100' to layer '154' of the base model along with the final added dense layer on the 'Sign Language Digit Dataset'. Therefore in this method, the number of trainable parameters is more as opposed to the above method. The following are the hyper-parameter values used for training the MobileNetV2 model using this method, 'RMS Prop optimizer' with a learning rate of 0.00001, batch size of 32 for a total of 25 epochs. The architecture and the total number of parameters used for training the MobileNetV2 model using this method are shown in Tables 2 and 3 respectively.

2.3.2. Proposed Model

We propose a lightweight CNN (as shown in Figure 4) for the task in hand. The proposed network utilizes a novel bottleneck motivated by deep residual learning [41]. The bottleneck consists of stacked residual blocks as presented in Figure 4 and Table 4. The layer-wise details like type of operation performed, size of feature maps of the overall architecture are shown in Table 5.

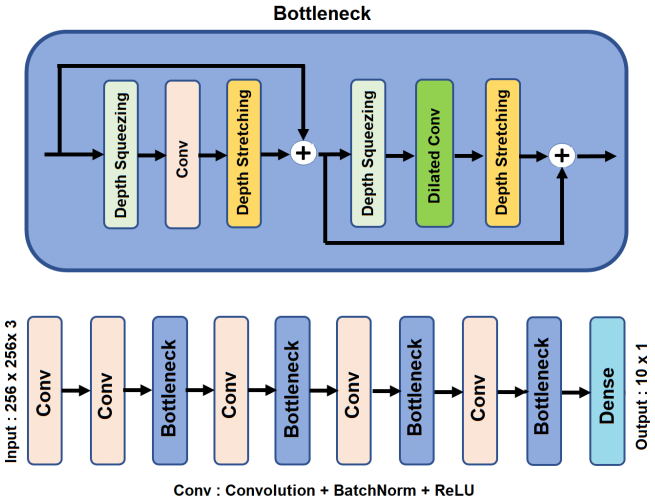


Figure 4. Flow diagram of the proposed deep learning model for hand gesture recognition. Layer wise details along with the sizes of the feature maps are shown in Table 5.

Table 4. Architecture details of the proposed Bottleneck within proposed CNN presented in Table 5. Note that 'M' is spatial extent and 'Z' is depth of the feature maps.

Layer Type	Operation	Output Size
Input	-	(None, M, M, Z)
1×1 Conv	Depth Squeezing	(None, M, M, Z/4)
3×3 Conv	Feature Extraction	(None, M, M, Z/4)
1×1 Conv	Depth Squeezing	(None, M, M, Z)
Add	Addition	(None, M, M, Z)
1×1 Conv	Depth Squeezing	(None, M, M, Z/4)
3×3 Dilated Conv	Feature Extraction	None, M, M, Z/4)
1×1 Conv	Depth Stretching	(None, M, M, Z)
Add	Addition	(None, M, M, Z)
Output	-	(None, M, M, Z)

Table 5. Architecture details of the proposed CNN for hand gesture recognition. Note that each Conv layer represents 3×3 convolution, followed by ReLU activation along with batch normalization. The flow of the architecture is from left to right in each row (top to bottom for successive step). The details of the bottleneck are given in Table 4.

Layer Type	Input	Conv	Conv	Bottleneck
Output Size	(None, 256, 256, 3)	(None, 127, 127, 64)	(None, 63, 63, 128)	(None, 63, 63, 128)
Layer Type	Conv	Bottleneck	Conv	Bottleneck
Output Size	(None, 31, 31, 256)	(None, 31, 31, 256)	(None, 15, 15, 128)	(None, 15, 15, 128)
Layer Type	Conv	Bottleneck	Conv	Flatten
Output Size	(None, 7, 7, 64)	(None, 7, 7, 64)	(None, 3, 3, 16)	(None, 144)
Layer Type	Dense	-	-	-
Output Size	(None, 10)	-	-	-

The initial residual block is motivated from Ref. [41] and it consists of 1×1 convolution for depth squeezing followed by 3×3 convolution for local feature extraction and finally 1×1 convolution for depth stretching. This kind of 1×1 projection-based embeddings possesses more information about large input patch [42]. The second residual block is similar to the first one except that the 3×3 convolution is performed in a dilated manner as shown in Figure 5 for improving the receptive field throughout the network. In total, the proposed CNN has 4 blocks of such bottlenecks arranged alternatively as

presented in Table 5 and each convolution operation in the proposed model is followed by ReLU [43] and batch normalization [44]. On comparing with MobilenetV2, the proposed model has 2.35 × fewer parameters and 1.85 × less memory usage (refer to Table 6). In short, the operations performed in the proposed bottleneck can be summarized as follows: Given a feature map x (at the input of bottleneck), the output $h(x)$ of the initial residual block can be written as:

$$h(x) = f(x, \theta_1) + x \tag{1}$$

where, $f(x, \theta)$ is sequence of convolution operations parameterized by θ_1, performing depth squeezing, feature extraction and depth stretching. Note that it is easy to optimize $f(x, \theta_1)$, than to learn the underlying $h(x)$ directly from x [41]. Finally, the output $b(x)$ of the bottleneck is given by:

$$b(x) = g(h(x), \theta_2) + h(x) \tag{2}$$

where, $g(x, \theta_2)$ is sequence of convolution operations parameterized by θ_2, performing depth squeezing, feature extraction via dilated convolution and depth stretching. Given a mini-batch with N samples, the cross-entropy loss $\mathcal{L}$ was computed as shown below:

$$\mathcal{L} = -\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \sum_{t=0}^{\mathrm{C}-1} y_{it} \log\left(y'_{it}\right) \tag{3}$$

where y is the one hot encoded label ($C \times 1$) and y' is the predicted softmax probabilities ($C \times 1$) and, C is number of classes. Being extremely lightweight, the proposed network does not require any transfer learning the model was trained end to end on Google Colab using Keras deep learning library [45]. We use the following hyper-parameter values to train the model, 'Adam optimizer' [46] with a learning rate of 0.005, weights initialised using kaiming initialization [47], batch size of 8 for a total of '40' epochs.

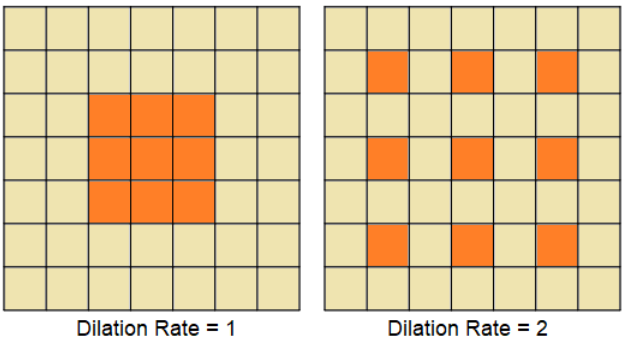


Figure 5. Sample example showing 3 × 3 convolution on 7 × 7 patch with dilation rates of 1 and 2 utilized in the proposed CNN.

2.4. Deployment on Edge Computing Device

A low-cost edge computing hardware, the Raspberry pi 4 model B microprocessor is utilized for implementing the given task. Raspberry pi 4 model B is the latest product from the raspberry pi collection of single-board edge computing devices. This new version of raspberry pi provides increased processor speed, connectivity, and memory capacity compared to the raspberry pi 3 model B+. The cost of this edge computing device is around $35. We attached a Raspberry pi camera V2.1 camera module to the raspberry pi 4 micro-controller to capture images in real-time. The camera module is capable of capturing images at 3280 × 2464 pixel resolution. It can also take videos at 1080p30, 720p60 and 640 × 480 p90 quality. The camera module costs around $27. The trained model is deployed on the raspberry pi 4 to predict the hand gestures. The entire Tensorflow version of the proposed and MobileNetV2 model consists of several parameters and using these models in its original version will cause latency while predicting. To tackle this latency issue for real-time applications, the Tensorflow-lite (TFL) version of these models are developed and

deployed. The inference time of both these models along with the model size is presented in Table 6. On comparing with MobilenetV2, the proposed model has 2.35 × less parameters and 1.85 × less memory usage (refer Table 6). The complete setup of the hardware along with a one Euro coin for size comparison has been shown in Figure 6.

Table 6. Comparison of Deep learning models considered in this work for deploying in the edge computing device (Raspberry pi 4 model B). Note that notation wise, TF represents Tensorflow and TFL represents' Tensorflow-Lite.

Model Tag	**MobileNetV2**	**Proposed**
Parameters	2.27 Million	0.97 Million
TF Model Size	11.1 MB	6 MB
TFL Model Size	8.5 MB	4 MB
TFL Inference time	212 ms	280 ms

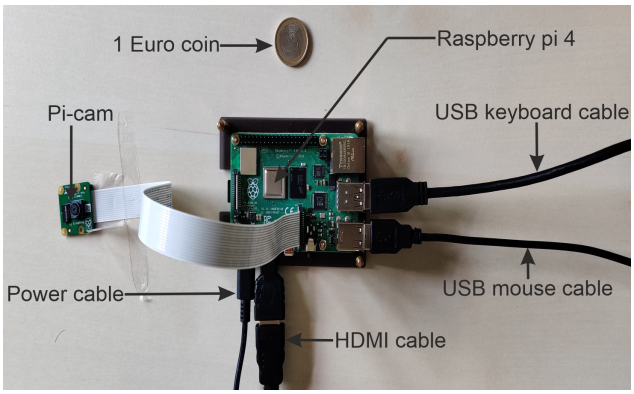


Figure 6. Complete hardware setup with labelled parts and size comparison with a one Euro coin.

During real-time testing, the images captured from the camera are the input to the system. These images are of the size [3280 × 2464] pixels are resized (downscaled) to [256 × 256] pixels before it is sent as input to the deep learning model. This is because, for real-time prediction, images whose resolution were below [256 × 256] were distorted. The entire workflow of the contactless authentication process has been shown in Figure 7. The variable 'i' shown in Figure 7 is the counter variable to keep track of the number of iterations that run in the system. We run the loop 'n' times, where n is the length of the authentication code. Inside the loop, the system performs two basic steps. In the first step, the system takes the real-time camera feed from the pi-cam as the input and stores the entire video (consisting of multiple image frames) in the frame buffer. In the next step, the input image (the first image frame in the buffer) is first resized to [256 × 256] pixel size and then trained proposed deep learning model classifies it into one of the ten-digit classes and this predicted digit class is printed on the screen. We then pause for 2 s during which the human changes the digit sign and also the frame buffer is emptied of all its stored frames. Next, we continue with the next iteration of the loop. This stoppage time for sign digit transition is programmable and can be adapted to the application requirements. Once the loop terminates, the authentication code is displayed on the screen. For verification in this work, it gets printed. As the digit is represented as ASCII code, it can directly authenticate any device (an example being ATM).

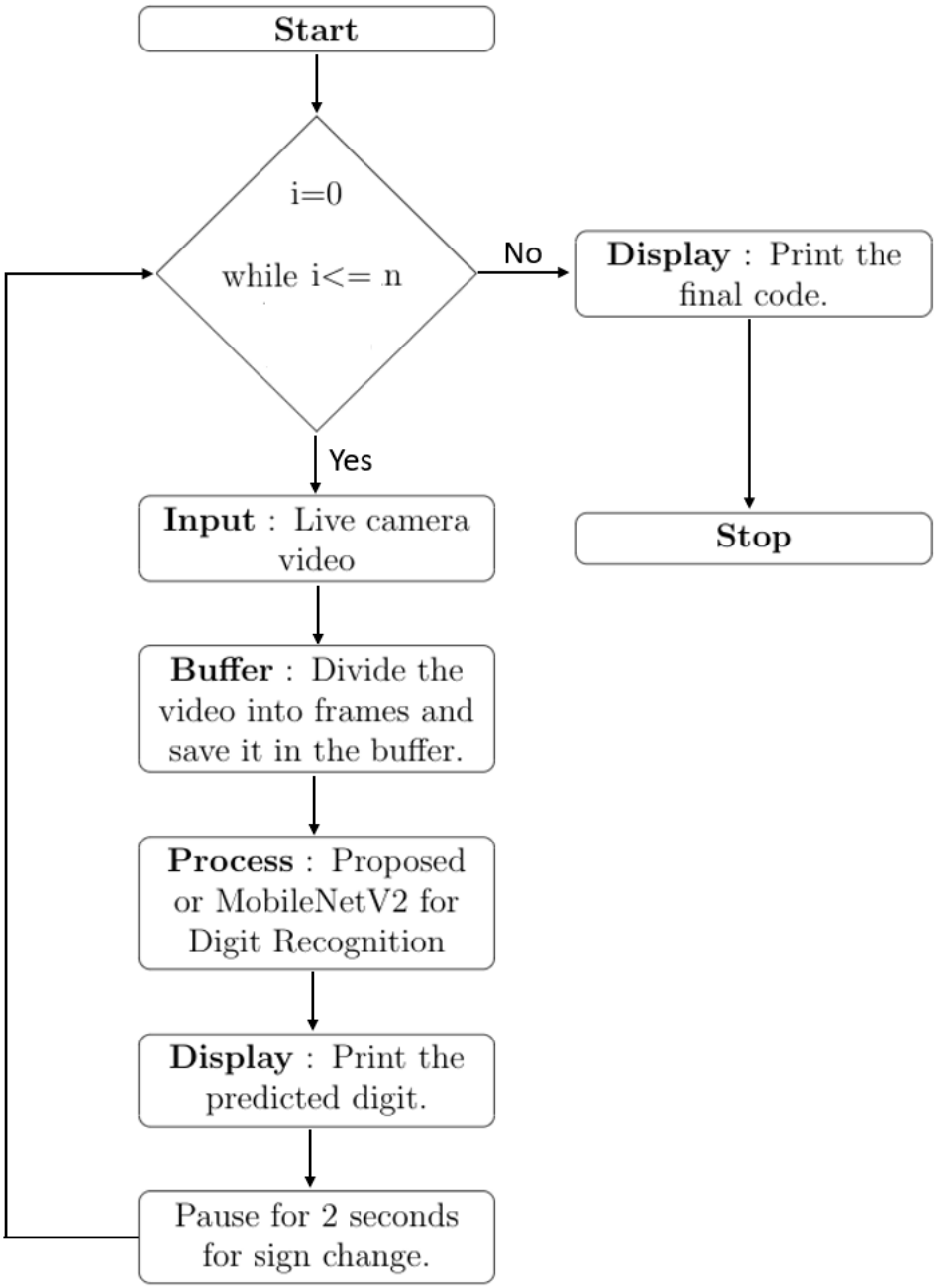


Figure 7. Workflow diagram for the contactless authentication code generation. Note that 'n' is the length of the code (n = 4 here).

2.5. Performance Evaluation Metrics

We use the following performance evaluation metrics to compare the proposed model and the MobileNetV2 model.

2.5.1. Confusion Matrix

This is used to describe the performance of a classifier in terms of a matrix. This matrix consists of 4 important elements namely 'True Positive (TP)', 'True Negative (TN)', 'False Positive (FP)' and 'False Negative (FN)' [48].

2.5.2. True Positive

True positive indicates the number of predictions where the model classifies correctly a positive sample into a positive class.

2.5.3. True Negative

True negative indicates the number of predictions where the model classifies correctly a negative sample into a negative class.

2.5.4. False Positive

False positive indicates the number of predictions where the model classifies incorrectly a negative sample into a positive class.

2.5.5. False Negative

False negative indicates the number of predictions where the model classifies incorrectly a positive sample into a negative class.

2.5.6. Accuracy

Accuracy indicates how well the 'TP' and 'TN' are predicted by the classifier [48]. It also tells the overall accuracy of a model. The metric is calculated according to the Equation (4).

$$\text{Accuracy} = (TP + TN)/(TP + FP + TN + FN) \tag{4}$$

2.5.7. Recall

This metric calculates the ratio of correctly predicted positive observations to all the observations that belong to the actual positive class [48]. This is calculated according to the Equation (5).

$$\text{Recall} = (TP)/(TP + FN) \tag{5}$$

2.5.8. Precision

This metric calculates the ratio of correctly predicted positive observations to all the predicted positive observations [48]. This is calculated according to the Equation (6).

$$\text{Precision} = (TP)/(TP + FP) \tag{6}$$

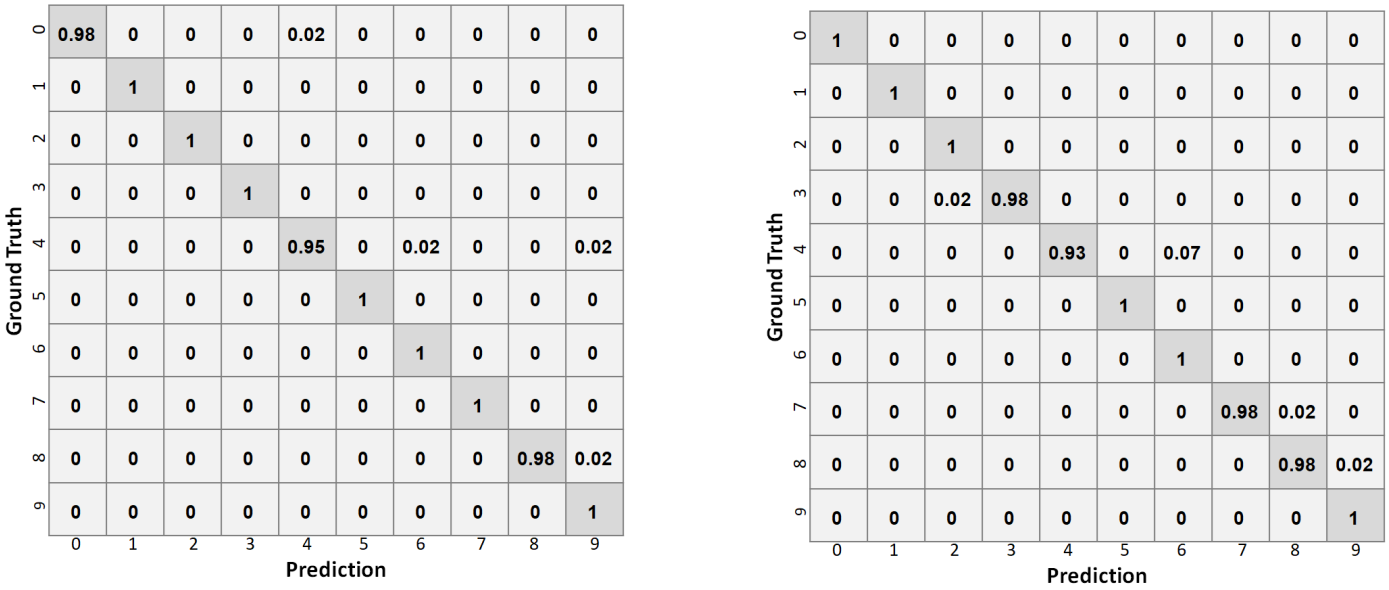
2.5.9. F1-Score

This metric uses both recall and precision to calculate its value [48]. It is a weighted average between the mentioned metrics according to the Equation (7)

$$\text{F1-score} = 2 \times (Recall \times Precision)/(Recall + Precision) \tag{7}$$

3. Results

The confusion matrices (on test set) for both the models are shown in Figure 8. It can be observed from Figure 8a that the proposed CNN model correctly predicts (True Positive) 100% of the test samples corresponding to the digits '1, '2', '3', '5', '6', '7' and '9'. Digits '0' and '8' are correctly predicted for 98% of the test samples corresponding to it and the digit '4' is correctly predicted for 95% of test samples corresponding to it. From Figure 8b, the MobilenetV2 model predicts correctly (True Positive) 100% of the test samples related to the digits '0', '1', '2', '5', '6' and '9'. Digits '3', '7' and '8' are predicted correctly for 98% of the test samples corresponding to it and digit '4' is predicted correctly for 93% of the test samples related to it.



(**a**) Proposed CNN Model. (**b**) MobileNetV2 Model.

Figure 8. Confusion matrix of both the models on test dataset.

Using the confusion matrices, all the other evaluation metrics are calculated. The 'Accuracy' metric for both the models is shown in Table 7. The other evaluation metrics for both the models are summarized in Table 8. As seen from Table 8, the 'Recall' metric for the proposed CNN model is better than the MobilenetV2 model for all the digits except digit '0'. The 'Precision' metric for the proposed CNN model is better than MobilenetV2 model especially for digits '2', '4', '6' and '9'. The 'F1-score' metric value for the proposed model is better than the MobilenetV2 model especially for the digits '0', '2', '3', '6' and '8'.

Table 7. Accuracy metric value for the proposed CNN model and MobileNetV2 model.

Model	Accuracy
Proposed CNN	99.1%
MobileNetV2	98.5%

Table 8. Recall, Precision and F1-score evaluation metric values for both the proposed model and MobileNetV2 model.

Class	Recall		Precision		F1-score	
	Proposed CNN	MobileNetV2	Proposed CNN	MobileNetV2	Proposed CNN	MobileNetV2
0	0.98	1.0	1.0	1.0	0.99	1.0
1	1.0	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	0.98	1.0	0.99
3	1.0	0.98	1.0	1.0	1.0	0.99
4	0.95	0.93	0.97	1.0	0.96	0.96
5	1.0	1.0	1.0	1.0	1.0	1.0
6	1.0	1.0	0.98	0.93	0.99	0.97
7	1.0	0.98	1.0	1.0	1.0	0.99
8	0.98	0.98	1.0	0.98	0.99	0.98
9	1.0	1.0	0.95	0.98	0.98	0.99

The Figure 9 shows few examples of predicted labels by the proposed model on the test data. The real-time predictions of the designed system under different lighting conditions is shown in Figure 10. As it can be observed that the model predicts the authentication code correctly under both uniform and non-uniform lighting conditions.

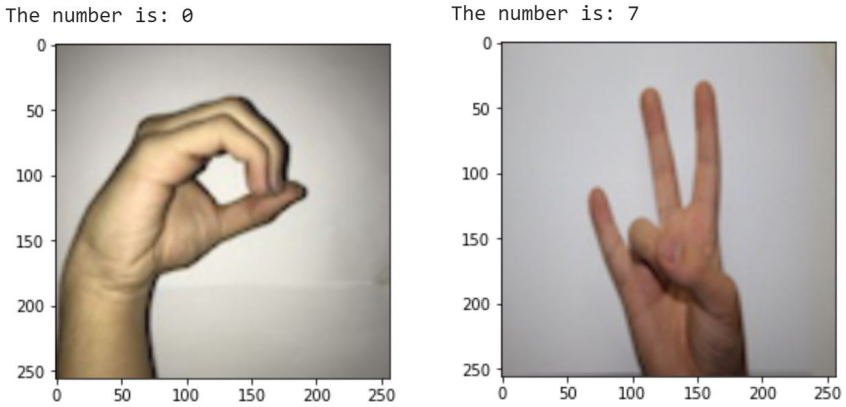


Figure 9. Predictions (given on top of images) of the proposed CNN in classifying the given test samples from the sign dataset.

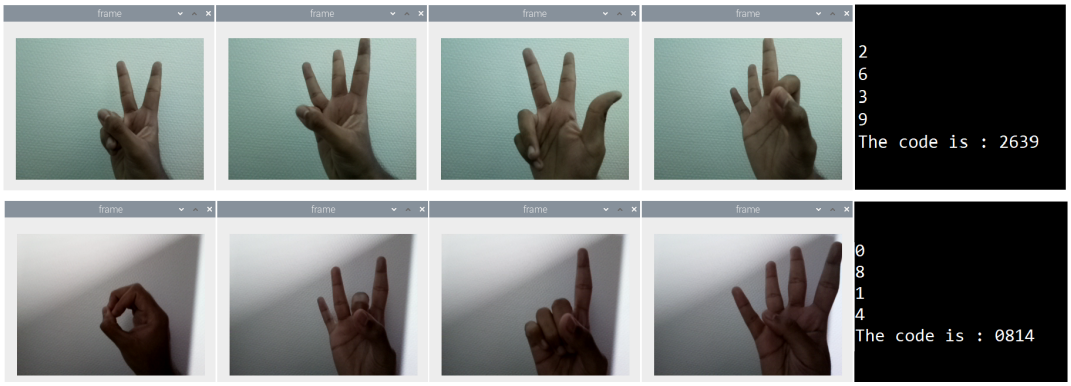


Figure 10. Four Numbers are predicted one after another in real-time under different natural lighting conditions (uniform lighting condition in first row and non uniform lighting condition in second row), where the corresponding sign language images were captured and processed using edge computing device deployed in this work, and the final code that was generated using proposed CNN given as the last image in each row.

4. Discussion

The results presented in this work indicate that contactless authentication based on hand gestures using a Raspberry pi-based system combined with deep learning is feasible. The proposed deep learning model is also compared to the popular MobileNetV2 and summarized in Table 6. It can be observed from Table 6 that the proposed model size is quite small compared to the MobileNetV2 model for both Tensorflow and Tensorflow-Lite. On the other hand, the proposed CNN model inference time is slightly higher than the MobileNetV2 due to the bottlenecks applied. Due to this, the computation pattern of compressed models becomes more irregular leading to severe data locality and load balancing issues, which in turn increase the inference/latency time.

One example use case for the developed system could be the generation of authentication PIN without touching the keypad. As both sides of a typical ATM usually have enclosures, the digit signs (hand gestures) will not be visible to others and the whole authentication can be a low-cost solution. As shown in Figure 10, the code generation is oblivious to lighting conditions and achieves good accuracy in both uniform and non-uniform lighting conditions. The Raspberry pi 4 model B utilized as an edge computing device in this work can provide various modes of output to feed the code for the security systems. The developed models, both Tensorflow and Tensorflow lite versions, are available as open-source for enthusiastic users at https://github.com/aveen-d/sign_detection.

5. Limitations

The number of samples of each class (0–9) in the utilized hand gesture sign recognition dataset is limited. However, the performance of the proposed deep learning model is promising as discussed in the results section. Also, the dataset with additional classes can further be enhanced in future work. The motion artifacts caused during image acquisition might slightly reduce the performance of the model. The limitation on the field of view of the camera and practical positioning of the hands in front of the camera may affect the performance.

6. Conclusions

This work presented the design of a complete end-to-end system, which can utilize the hand gestures of the sign language digits for authenticating the user contactlessly at public and business places, such as ATMs, gas stations, and shopping malls. To be more specific, we have developed a convolutional neural network (CNN) to provide authentication code based on camera data, making it truly contactless. The whole inference of the deep learning model including data capture, converting imaging data to an authentication code was performed on a Raspberry pi 4 model B microprocessor coupled with a camera (total cost being less than $75) attached to it to serve as a complete solution, making it highly suitable for large scale deployment. The designed system achieved an accuracy of 99.1% on the test data set compared to the popular MobileNetV2 model, whose accuracy is 98.5%, despite proposed CNN being 2.35 $\times$ lighter in terms of the number of parameters. The developed system works in real-time with the model inference rate of 280 ms per image frame and the designed system can be an alternative solution for the conventional authentication methods, which use touchpads and keypads. In the future, we can increase the dataset with different classes such as “alphabets”, “accept”, “close”, and “go back” to further reduce the need for contact with the surface for navigating through a web page/applications in these secure systems.

Author Contributions: Conceptualization, L.R.C.; Data curation, A.D.; Formal analysis, N.P.; Methodology, A.D., N.P., L.R.C., S.J. and P.K.Y.; Writing—original draft, A.D. and L.R.C.; Writing—review & editing, N.P., S.J. and P.K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Indo-Norwegian Collaboration in Autonomous Cyber-Physical Systems (INCAPS) project: 287918 of the International Partnerships for Excellent Education, Research and Innovation (INTPART) program from the Research Council of Norway.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs. Publicly available dataset was utilized in this study. The developed models, both Tensorflow and Tensorflow lite versions, are available as open-source for users https://github.com/aveen-d/sign_detection. The ‘Sign Language Digits Dataset’ is available https://github.com/ardamavi/Sign-Language-Digits-Dataset.

Acknowledgments: We would like to thank the Norwegian Research Council for the support through the INCAPS project: 287918 of the INTPART program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jain, A.K.; Ross, A.; Prabhakar, S. An introduction to biometric recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2004**, *14*, 4–20. doi:10.1109/TCSVT.2003.818349. [CrossRef]
2. Yahya, F.; Nasir, H.; Kadir, K.; Safie, S.; Khan, S.; Gunawan, T. Fingerprint Biometric Systems. *Trends Bioinform.* **2016**, *9*, 52–58. doi:10.3923/tb.2016.52.58. [CrossRef]
3. Zhao, W.; Chellappa, R.; Phillips, P.J.; Rosenfeld, A. Face Recognition: A Literature Survey. *ACM Comput. Surv.* **2003**, *35*, 399–458, doi:10.1145/954339.954342. [CrossRef]
4. Zhang, D.; Lu, G.; Kong, A.W.K.; Wong, M. Palmprint Authentication System for Civil Applications. In *Biometric Authentication*; Maltoni, D., Jain, A.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 217–228.

5. Mazumdar, J. Retina Based Biometric Authentication System: A Review. *Int. J. Adv. Res. Comput. Sci.* **2018**, *9*, 711–718. doi:10.26483/ijarcs.v9i1.5322. [CrossRef]
6. Chowdhury, R.; Ghosh, D.; Agarwal, P.; Samir, K.; Bandyopadhyay, S. Ear Based Biometric Authentication System. *World J. Eng. Res. Technol.* **2018**, *2*, 224–233. Available online: https://www.researchgate.net/publication/328610283_EAR_BASED_BIOMETRIC_AUTHENTICATION_SYSTEM (accessd on 12 December 2020).
7. Olatinwo, S.; Shoewu, O.; Omitola, O. Iris Recognition Technology: Implementation, Application, and Security Consideration. *Pac. J. Sci. Technol.* **2013**, *14*, 228–233.
8. Ali, M.; Tappert, C.; Qiu, M.; Monaco, V. Keystroke Biometric Systems for User Authentication. *J. Signal Process. Syst.* **2017**, *86*, doi:10.1007/s11265-016-1114-9. [CrossRef]
9. Jabin, S.; Zareen, F. Biometric signature verification. *Int. J. Biom.* **2015**, *7*, 97, doi:10.1504/IJBM.2015.070924. [CrossRef]
10. Gafurov, D. A Survey of Biometric Gait Recognition: Approaches, Security and Challenges. In *Proceedings of the Norsk Informatikkonferanse*; Oslo, Norway, 2007. Available online: https://www.researchgate.net/profile/Davrondzhon_Gafurov/publication/228577046_A_survey_of_biometric_gait_recognition_Approaches_security_and_challenges/links/00b49528e834aa68eb000000.pdf (accessed on 12 December 2020).
11. Iannizzotto, G.; Rosa, F. A SIFT-Based Fingerprint Verification System Using Cellular Neural Networks. In *Pattern Recognition Techniques, Technology and Applications;* I-Tech: Vienna, Asutria, 2008. [CrossRef]
12. Naidu, S.; Chemudu, S.; Satyanarayana, V.; Pillem, R.; Hanuma, K.; Naresh, B.; CH.Himabin, d. New Palm Print Authentication System by Using Wavelet Based Method. *Signal Image Process.* **2011**, *2*, doi:10.5121/sipij.2011.2114. [CrossRef]
13. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. doi:10.1016/j.patcog.2017.10.013. [CrossRef]
14. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
15. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* **2017**, *105*. [CrossRef]
16. Zulfiqar, M.; Syed, F.; Khan, M.; Khurshid, K. Deep Face Recognition for Biometric Authentication. In Proceedings of the 2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Swat, Pakistan, 24–25 July 2019; doi:10.1109/ICECCE47252.2019.8940725. [CrossRef]
17. Praseetha, V.; Bayezeed, S.; Vadivel, S. Secure Fingerprint Authentication Using Deep Learning and Minutiae Verification. *J. Intell. Syst.* **2019**, *29*. [CrossRef]
18. Shao, H.; Zhong, D. Few-shot palmprint recognition via graph neural networks. *Electron. Lett.* **2019**, *55*, 890–892. [CrossRef]
19. Aizat, K.; Mohamed, O.; Orken, M.; Ainur, A.; Zhumazhanov, B. Identification and authentication of user voice using DNN features and i-vector. *Cogent Eng.* **2020**, *7*, 1751557. [CrossRef]
20. Terrier, P. Gait Recognition via Deep Learning of the Center-of-Pressure Trajectory. *Appl. Sci.* **2020**, *10*, 774. [CrossRef]
21. Poddar, J.; Parikh, V.; Bharti, D. Offline Signature Recognition and Forgery Detection using Deep Learning. *Procedia Comput. Sci.* **2020**, *170*, 610–617. [CrossRef]
22. Minaee, S.; Abdolrashidi, A.; Su, H.; Bennamoun, M.; Zhang, D. Biometric Recognition Using Deep Learning: A Survey. *arXiv* **2019**, arXiv:cs:CV/1912.00271.
23. Wu, K.; Chen, Z.; Li, W. A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks. *IEEE Access* **2018**, *6*, 50850–50859. [CrossRef]
24. Kim, K.H.; Hong, S.; Roh, B.; Cheon, Y.; Park, M. Pvanet: Deep but lightweight neural networks for real-time object detection. *arXiv* **2016**, arXiv:1608.08021.
25. Deng, B.L.; Li, G.; Han, S.; Shi, L.; Xie, Y. Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proc. IEEE* **2020**, *108*, 485–532. [CrossRef]
26. Dey, N.; Mishra, G.; Kar, J.; Chakraborty, S.; Nath, S. A Survey of Image Classification Methods and Techniques. In Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kanyakumaru, India, 10–11 July 2014; [CrossRef]
27. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. (IJCV)* **2015**, *115*, 211–252. [CrossRef]
28. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Master's Thesis, Department of Computer Science, Univ. Tor. **2009**. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 13 December 2020).
29. Parkhi, O.M.; Vedaldi, A.; Zisserman, A.; Jawahar, C.V. Cats and Dogs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012.
30. Nilsback, M.E.; Zisserman, A. A Visual Vocabulary for Flower Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, USA, 17–22 June 2006; Volume 2, pp. 1447–1454.
31. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 3 January 2021).
32. Foret, P.; Kleiner, A.; Mobahi, H.; Neyshabur, B. Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv* **2020**, arXiv:cs.LG/2010.01412.

33. Wang, X.; Kihara, D.; Luo, J.; Qi, G.J. EnAET: Self-Trained Ensemble AutoEncoding Transformations for Semi-Supervised Learning. *arXiv* **2019**, arXiv:cs.CV/1911.09265.
34. Kolesnikov, A.I.; Beyer, L.; Zhai, X.; Puigcerver, J.; Yung, J.; Gelly, S.; Houlsby, N. Big Transfer (BiT): General Visual Representation Learning. *Comput. Vis. Pattern Recognit.* **2019**, *6*, 8.
35. Byerly, A.; Kalganova, T.; Dear, I. A Branching and Merging Convolutional Network with Homogeneous Filter Capsules. *arXiv* **2020**, arXiv:abs/2001.09136.
36. Zeynep Dikle, A.M.; Students, T.A.A.A.H.S. Sign Language Digits Dataset. 2017. Available online: https://github.com/ardamavi/Sign-Language-Digits-Dataset (accessed on 2 June 2020).
37. Gavade, A.; Sane, P. Super Resolution Image Reconstruction By Using Bicubic Interpolation. In Proceedings of the ATEES 2014 National Conference, Belgaum, India, October 2013.10.13140/RG.2.1.4909.0401. [CrossRef]
38. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
39. Lin, M.; Chen, Q.; Yan, S. Network in Network. *arxiv* **2013**, arXiv:cs:CV/1312.4400.
40. Tieleman, T.; Hinton, G. Lecture 6.5—RmsProp: Divide the Gradient by a Running Average of Its Recent Magnitude. COURSERA: Neural Networks for Machine Learning, 2012. Available online: http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf (accessed on 4 January 2021).
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
42. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
43. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
44. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
45. Keras: The Python Deep Learning Library. Available online: http://ascl.net/1806.022 (accessed on 4 January 2021).
46. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1026–1034.
48. Ghori, K.M.; Abbasi, R.A.; Awais, M.; Imran, M.; Ullah, A.; Szathmary, L. Performance Analysis of Different Types of Machine Learning Classifiers for Non-Technical Loss Detection. *IEEE Access* **2020**, *8*, 16033–16048. [CrossRef]