Talking With Your Hands: Scaling Hand Gestures and Recognition With CNNs

Okan Köpüklü[1], Yao Rong[1,2], Gerhard Rigoll[1]

[1] Institute for Human-Machine Communication, TU Munich, Germany

[2] Infineon Technologies AG, Germany

Abstract

The use of hand gestures provides a natural alternative to cumbersome interface devices for Human-Computer Interaction (HCI) systems. As the technology advances and communication between humans and machines becomes more complex, HCI systems should also be scaled accordingly in order to accommodate the introduced complexities. In this paper, we propose a methodology to scale hand gestures by forming them with predefined gesture-phonemes, and a convolutional neural network (CNN) based framework to recognize hand gestures by learning only their constituents of gesture-phonemes. The total number of possible hand gestures can be increased exponentially by increasing the number of used gesture-phonemes. For this objective, we introduce a new benchmark dataset named Scaled Hand Gestures Dataset (SHGD) with only gesture-phonemes in its training set and 3-tuples gestures in the test set. In our experimental analysis, we achieve to recognize hand gestures containing one and three gesture-phonemes with an accuracy of 98.47% (in 15 classes) and 94.69% (in 810 classes), respectively. Our dataset, code and pretrained models are publicly available [1].

1. Introduction

Computers have become an indispensable part of human life. Therefore, facilitating natural human-computer interaction (HCI) contains utmost importance to bridge human-computer barrier. Gestures have long been considered as an interaction technique delivering natural and intuitive experience while communicating with computers. This is a driving force in the research community to work on gesture representations, recognition techniques and frameworks.

As technology keeps advancing, the use of computers in our lives increases as well with additional new devices such as smart phones, watches, TVs, headphones, autonomous cars etc. Therefore, the communication between humans and machines gradually becomes more complex, requiring HCI systems to accommodate the introduced complexities.

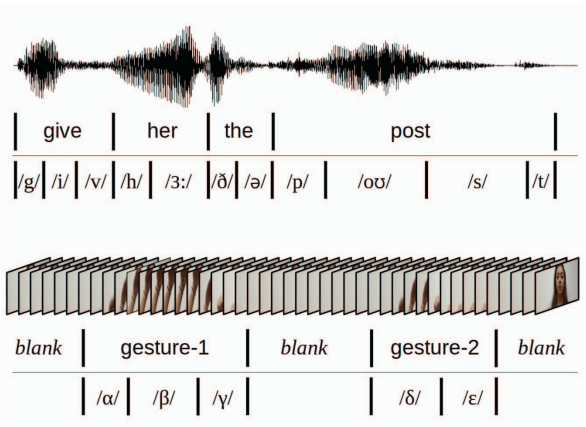


Figure 1: Top: An audio signal corresponding to the sentence "give her the post". Each word in this sentence consists of one or multiple phonemes. Bottom: A video signal (i.e. sequence of frames) containing 2 hand gestures. Similar to speech signal, each gesture consists of one or multiple gesture-phonemes denoted by $\alpha, \beta, \gamma, \delta, \varepsilon$. The signals and their annotations are for illustrative purposes only.

In this work, we propose an approach to scale hand gestures by composing each gesture with multiple gesture-phonemes. The main inspiration comes from the phonology and morphology of the spoken languages. Fig. 1 (top) shows the morphological and phonological analysis of the sentence "give her the post". Each word in this sentence is composed of a sequence of phonemes. Similarly, we create hand gestures using one or multiple gesture-phonemes sequentially, as shown in Fig. 1 (bottom). So, our motivation is first to learn the gesture-phonemes successfully, then to recognize hand gestures, which contains multiple gesture-phonemes, with only this knowledge.

Structuring hand gestures with this approach enables to scale hand gestures without requiring to collect additional training data. For a given number of gesture-phonemes, the number of all possible hand gestures is exponentially proportional to the number of gesture-phonemes each gesture contains. For the proposed gesture scaling approach,

[1] https://www.mmk.ei.tum.de/shgd/

we present a convolutional neural network (CNN) based framework using sliding-window approach together with Viterbi-like [29] decoder algorithm. For the CNN model, we have used 2-dimensional (2D) and 3-dimensional (3D) SqueezeNet and MobileNetV2 models.

This paper presents the following contributions:

(i) Our major contribution is creating hand gesture recognition framework, which is "scalable" according to the complexity of the desired HCI system. To the best of our knowledge, this is the first work that address the scalability of hand gestures. The CNN model is only trained with 10 gesture phonemes and 3 signaling classes (*preparation*, *retraction* and *no-gesture*), and the framework can recognize scaled gesture tuples with 3 gesture phonemes (as in this paper) or more. Assumed that a HCI system with the recognition capability of 810 different gestures needs to be implemented. With the old fashioned way, you need to define 810 different hand gestures, collect enough training samples (400 training samples for each class), train an architecture to get desired accuracy (remember that for ChaLearn IsoGD [30], the state-of-the-art accuracy is around 80% for 249 classes). With this framework, you just need to train with 10 gesture phonemes and 3 signaling classes, then for 810 classes (3-tuple gestures) you can achieve around 95% accuracy. Consider the situation when you need 65610 different gestures (5-tuple gestures). Approximately 25 million training samples are needed for the old fashioned way.

(ii) The second contribution is the benchmark dataset named Scaled Hand Gestures Dataset (SHGD), which will be made publicly available. The videos are collected using a Time-of-Flight (ToF) based 3D Image Sensor, which is shown in Fig. 2. The dataset contains only gesture-phonemes in its training set. For the test set, SHGD contains gesture-phonemes and 3-tuple gestures.

(iii) The third contribution of the paper is that with the designed Viterbi-like decoder, the performed 3-tuple gestures are recognized only once. This contains utmost importance for online HCI systems. Moreover, designed Viterbi-like decoder is very lightweight as HCI systems should be designed considering the memory and power budget of the HCI system.

2. Related Work

Ever since AlexNet [17], deep CNNs have dominated nearly all computer vision tasks. At first, CNNs have infiltrated to the image-based tasks due to the availability of only large scale image datasets such as ImageNet [3]. Afterwards, CNNs are also applied for video analysis tasks. However, as the first video datasets were comparatively small such as UCF-101 [28] and HMDB [18], all initial video analysis architectures are based on 2D CNNs which utilize transfer learning from ImageNet, such as [27, 14, 31, 4]. With the availability of large-scale video datasets like Sports-1M [14], Kinetics [1], Jester [7], this problem was solved and successful 3D CNNs could be trained from scratch without overfitting [9].

Since gestures provide a natural, creative and intuitive interaction experience for communication with computers, hand gesture recognition is one of the most popular video analysis tasks. Although there have been many approaches using hand-crafted features like orientation of histograms [5], histogram of oriented gradients (HOG) [25] or bag-of-features [2], the state of the art hand gesture recognition architectures are based on CNNs [16, 22, 21, 23, 15], similar to other computer vision tasks.

Until recently, the primary trend has been to make CNNs deeper and more complicated [12, 10] in order to achieve higher classification performance. But the pursue of lightweight networks with high accuracy is now growing, as in many real-time applications like autonomous driving and robotics, where the computation capability of the platform is always limited. Therefore, there has been several resource efficient CNN architectures such as SqueezeNet[13], MobileNet [11], MobileNetV2 [26], ShuffleNet [32] and ShuffleNetV2 [19], which aim to reduce computational cost but still keep the accuracy high. In our work, we have used the 2D and 3D versions of SqueezeNet and MobileNetV2 since we want a lightweight framework.

Fusion of different modalities is another strategy that helps CNNs to improve recognition performance. However, fusion also introduces extra computational cost especially at decision [27] and feature [20] level. On the other hand, [16] proposes a data level fusion strategy, Motion Fused Frames (MFFs), where different modalities can be fused with very little modification to the network and computational cost. Since we have infrared (IR) and depth modalities in our dataset, we have adapted data level fusion strategy.

Although there have been many gesture recognition approaches, the idea of scaling hand gestures is very new but also very important in order to create complex HCI systems. To the best of our knowledge, this is the first work that scales hand gestures. More importantly, besides scaling, we achieve very similar recognition performance for gesture-tuples (94.69% accuracy for 810 classes) compared to single gestures (98.47% accuracy for 15 classes).

3. Methodology

In this section, we fist describe the collected dataset. Afterwards, we explain the details of the experimented framework with its 2D and 3D CNN architectures and Viterbi-like decoder. Finally, we give the training details.

Figure 2: Data collection setup. Dataset is collected in infrared (bottom-left) and depth (bottom-right) modalities using Infineon® IRS1125C REAL3™ 3D Image Sensor.

3.1. Scaled Hand Gestures Dataset (SHGD)

SHGD contains 15 single hand gestures, each recorded for infrared (IR) and depth modalities using Infineon® IRS1125C REAL3™ 3D Image Sensor. Each recording contains 15 gesture samples (one sample per class). There are in total 324 recordings from 27 distinct subjects in the dataset. Recordings of 8 subjects are reserved for testing, which makes 30% of the dataset. Every subject makes 12 video recordings using two hands under 6 different environments, which are designed for increasing the network robustness against different lightning conditions and background disturbances. These environments are (1) indoors under normal daylight, (2) indoors under daylight and with an extra person in the background, (3) indoors at night under artificial lighting, (4) indoors in total darkness, (5) outdoors under intense sunlight and (6) outdoors under normal sunlight. We have simulated outdoor environments using two bright lights: Two lights for "intense sunlight" and one light for "normal sunlight".

Fig. 2 shows data collection setup, used camera and data samples. Subjects performed gestures while observing the computer screen, where the gestures were prompted in a random order. Videos are recorded at 45 frames per second (fps) with spatial resolution of 352×287 pixels. Each recording lasts around 33 seconds.

3.1.1 Single Gestures

In its training set, SHGD contains only single gestures under 15 classes, which are given in Table 1. Recordings in the dataset are continuous video streams meaning that each recording contains *no-gesture* and *gesture* parts. Moreover,

Label	Gesture	Label	Gesture	Label	Gesture
1	Fist	6	Two Fingers	11	Swipe Left*
2	Flat Hand	7	Five Fingers	12	Swipe Right*
3	Thumb Up	8	Stop Sign	13	Pull Hand In*
4	Thumb Left	9	Check	14	Move Hand Up*
5	Thumb Right	10	Zero	15	Move Hand Down*

Table 1: 15 single gesture classes in Scaled Hand Gesture Dataset (SHGD). * marks the dynamic gestures which are not included as gesture-phonemes.

each *gesture* contains *preparation*, *nucleus* and *retraction* phases [24, 6, 8], which are critical for real-time gesture recognition.

Among the single gesture classes listed in Table 1, static gestures are selected as gesture-phonemes since it is more convenient to perform different static gestures sequentially. For the rest of the paper, we will use the term *phoneme* instead of *gesture-phoneme* for the sake of easiness.

3.1.2 Gesture Tuples

Gesture tuple refers to hand gestures which contain sequentially performed phonemes. There are in total 10 different phonemes. When constructing gesture tuples, we leave out the consecutive same phonemes to avoid sequence length confusion. Therefore, the total number of different tuples can be calculated by the following equation:

$$N = m(m-1)^{(s-1)} \tag{1}$$

where m is the number different phonemes and s is the number of phonemes that the gesture tuple contains.

Besides the test set for single gestures, SHGD also has a test set for gesture tuples containing 3 phonemes. 5 subjects perform gesture tuples under 5 different lightning conditions (excluding the environment of (2)). There are in total $10\times(10-1)^{(3-1)} = 810$ permutations meaning different classes for 3-tuple gestures. Recordings are not segmented for this case. Therefore, one recording contains *no-gesture*, *3-tuple gesture* and *no-gesture* without exact location of *3-tuple gesture*.

Since gestures are performed at different speeds in the real-life scenarios, we have also collected 3-tuple gestures at three different speeds: Slow, medium and fast. The subjects should finish 3-tuple gestures within 300 frames (6.7 sec), 240 frames (5.3 sec) and 180 frames (4 sec) for slow, medium and fast speed, respectively.

3.1.3 SHGD-15 and SHGD-13

SHGD-15 refers to the standard dataset where all single gestures in Table 1 are included. On the other hand, SHGD-13 is specifically designed for 3-tuple gesture recognition. Besides 10 phonemes, SHGD-13 also contains *preparation*

(raising hand), *retraction* (lowering hand) and *no-gesture* classes. As there is no indication when a gesture starts and ends in the video, we use *preparation* and *retraction* classes to detect Start-of-Gesture (SoG) and End-of-Gesture (EoG). We use *no-gesture* class to reduce the number false alarms since most of the time, "no gesture" is performed in real-time gesture recognition applications [15].

SHGD-15 is a balanced dataset with 96 samples in each class. However, SHGD-13 is an imbalanced dataset, where *preparation* and *retraction* classes contain 10 times more samples than phonemes, whereas *no-gesture* contains around 20 times more samples than phonemes. Therefore, training of SHGD-13 requires special attention.

3.2. Network Architecture

The general workflow of the proposed architecture is depicted in Fig. 3. A sliding window goes through the video stream with a queue size of 8 frames and stride s of 1. The frames in the input queue is passed to a 2D/3D CNN which is pretrained on SHGD-13. The classification results are then post-processed by averaging with non-overlapping window size of 5. In this way, we can filter out some fluctuations due to the ambiguous states while changing the phonemes. Next, the post-processed outputs are fed into a detector queue, which tries to detect SoG and EoG. When the sum of class scores for *preparation* is higher than the threshold, we set SoG flag on, activate the classifier queue and start storing the post-processed scores. Then, the detector queue is responsible for detecting EoG in a similar manner. After EoG flag is received, we deactivate the classifier queue and run the Viterbi-like decoder which recognizes the 3-tuple gesture. In the next parts, we explain the details for the main building blocks in the proposed architecture.

3.2.1 2D and 3D CNN Classifiers

CNN classifier is the most critical part of the proposed architecture. The properties of the deployed CNNs determine the detection and classification performance, memory usage and the speed of the overall architecture. In order to fulfill the resource constrained conditions and run as a real time application, two lightweight models are preferred selecting SqueezeNet [13] and MobileNetV2 [26] as classifiers in our architecture. In our analyses, we have deployed the 2D and 3D versions of these models.

The input to the CNN classifier is always 8 frames. Using these 8 frames, CNN classifier should recognize static phonemes together with dynamic preperation and retraction classes successfully. 3D CNNs can capture this dynamic motion information inherently due to their 3D convolutional kernels. However, 2D CNNs requires an extra spatiotemporal modeling in order to reason the relations between different frames.

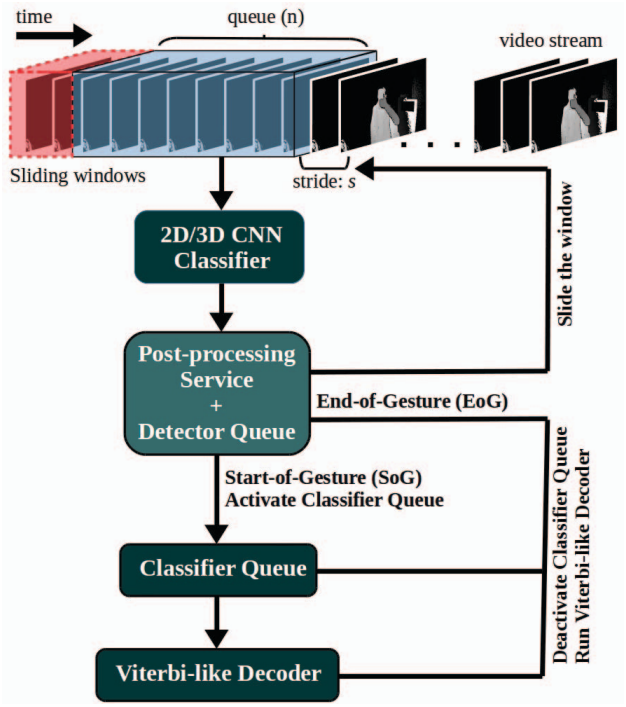


Figure 3: The general workflow of the proposed architecture. Sliding windows with stride s run through incoming video frames, and these frames in the queue are fed to a 2D or 3D CNN based classifier. The classifier's results are post-processed afterwards. After Start-of-Gesture (SoG) gets detected, the classifier queue is activated. Classifier's results are saved in the classifier queue until End-of-Gesture (EoG) is detected. Then, the Viterbi-like decoder runs on the classifier's queue to recognize the 3-tuple gesture.

Fig. 5 depicts the applied spatiotemporal modeling approach used for 2D CNN models. Features of each 8 frames are extracted using the same 2D CNN and concatenated keeping their order intact. Afterwards, two levels of fully connected (fc) layers are applied in order to get class-conditional probability scores. The reason behind is that fc layers can organically infer the temporal relations, without knowing it is a sequence at all. The size of features 2D CNNs extracts is 64 for each frame. With the first fc layer, feature dimension is reduced from 64×8=512 to 256. With the second fc layer, dimension is reduced to the number of classes.

On the other hand, 3D CNNs contains spatiotemporal modeling intrinsically and does not require an extra mechanism. We have inflated SqueezeNet and MobileNetV2 such that they accept 8 frames as input. The details of the 3D-SqueezeNet and 3D-MobileNetV2 are given in Table 2 and Table 3, respectively. Their main building blocks are also depicted in Fig. 4.

3D-SqueezeNet is deployed with simple bypass, as it

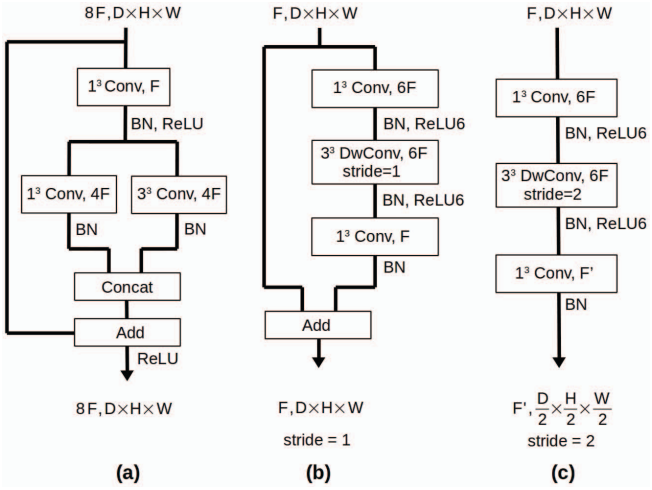


Figure 4: Blocks used in 3D CNN architectures. F is the number of feature maps and D×H×W stands for Depth×Height×Width for the input and output volumes. DwConv stands for depthwise convolution. 1^3 and 3^3 refers to kernel sizes of 1×1×1 and 3×3×3, respectively. **(a)** SqueezeNet's Fire block with simple bypass; **(b)** MobileNetV2's inverted residual block with stride 1; **(c)** MobileNetV2's inverted residual block with spatiotemporal downsampling (2×).

achieves better results in the original architecture. However, we have not used simple bypass for its 2D version, as 2D-SqueezeNet pretrained on ImageNet is only available without bypass. For MobileNetV2, we have used $width\_multiplier$ of 1 for both 2D and 3D versions.

The spatial size of the inputs are 224 and 112 for 2D and 3D CNNs, respectively. The number of input channels c depends on the experimented input data modality. Besides IR and depth, we have also applied data level fusion to IR and Depth (IR+D) in our experiments. We have used RGB modality only in pretrainings. Accordingly, the

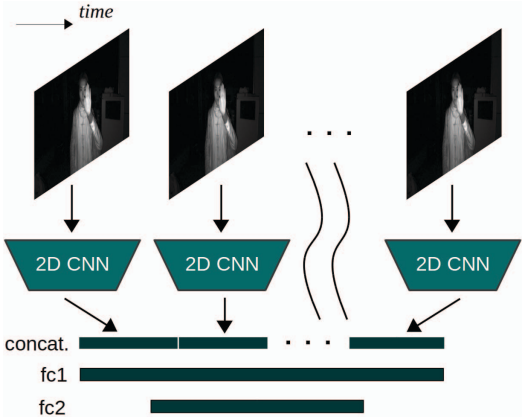


Figure 5: Spatiotemporal modeling approach used for 2D CNN models.

Layer / Stride	Filter size	Output size
Input clip		c×8×112×112
Conv1/s(1,2,2)	3×3×3	64×8×56×56
MaxPool/s(1,2,2)	3×3×3	64×8×28×28
Fire2		128×8×28×28
Fire3		128×8×28×28
MaxPool/s(2,2,2)	3×3×3	128×4×14×14
Fire4		256×4×14×14
Fire5		256×4×14×14
MaxPool/s(2,2,2)	3×3×3	256×2×7×7
Fire6		384×2×7×7
Fire7		384×2×7×7
MaxPool/s(2,2,2)	3×3×3	384×1×4×4
Fire8		512×1×4×4
Fire9		512×1×4×4
Conv10/s(1,1,1)	1×1×1	*NumCls*×1×4×4
AvgPool/s(1,1,1)	1×4×4	*NumCls*

Table 2: 3D-SqueezeNet architecture. Fire block is depicted in Fig. 4 (a).

number of input channels are 3, 2, 1, 1 for RGB, IR+D, IR, depth modalities, respectively. The final size of inputs are c×224×224 for 2D CNNs, and c×8×112×112 for 3D CNNs.

3.2.2 Viterbi-like Decoder

Viterbi decoding was invented by Andrew Viterbi [29] and is now widely used in decoding convolutional codes. It is an elegant and efficient way to find out the optimal path with minimal error. In this paper, we have adapted it and used a Viterbi-like decoder to find out the phoneme sequences in

Layer / Stride	Repeat	Output size
Input clip		c×8×112×112
Conv1(3×3×3)/s(1,2,2)	1	32×8×56×56
Block/s(1,1,1)	1	16×8×56×56
Block/s(1,2,2)	2	24×8×28×28
Block/s(2,2,2)	3	32×4×14×14
Block/s(2,2,2)	4	64×2×7×7
Block/s(1,1,1)	3	96×2×7×7
Block/s(2,2,2)	3	160×1×1×1
Block/s(1,1,1)	1	320×1×1×1
Conv(1×1×1)/s(1,1,1)	1	1280×1×1×1
Linear(1280×*NumCls*)	1	*NumCls*

Table 3: 3D-MobileNetV2 architecture. Block is inverted residual block whose details are given in Fig. 4 (b) and (c). Expansion factor of 6 is applied except for the initial Block where expansion factor of 1 is applied.

3-tuple gestures with maximal probability. Same as conventional Viterbi algorithm, we narrow down the optional paths systematically for each new input in the classifier queue.

For the Viterbi-like decoder, we introduced a couple of terms for better comprehensibility: *K* is the number of allowed state transitions in the output sequence, which is 2 as we use 3-tuple gestures. The state refers to a phoneme in a path for the given time instant. *P* refers to class-conditional probability scores for phonemes stored in Classifier Queue, which is shown in (2), whose columns P_t are the average probability scores of each phoneme for five consecutive time instants. P_t values are softmaxed before putting in *P*. *T* is the length of *P* (i.e. number of columns), and *N* is the number of phoneme classes, which is 10 in our case. Therefore, the size of *P* is $T \times N$.

$$P = \begin{bmatrix} | & \cdots & | & \cdots & | \\ P_0 & \cdots & P_t & \cdots & P_{T-1} \\ | & \cdots & | & \cdots & | \end{bmatrix}, \; P_t = \begin{bmatrix} p_{t,0} \\ p_{t,1} \\ \vdots \\ p_{t,N-1} \end{bmatrix} \quad (2)$$

The probability of a path is the sum of the probability scores of all the states that this path goes through. Besides the number of allowed transitions *K*, we introduce another constraint, transition cost δ, in order to prevent false state transitions in the path. A path metric *M* holds the paths $m_{t,i}$ with their sequence record $\pi_{t,i}$, path score $s_{t,i}$ and the transition times $k_{t,i}$. The path $m_{t,i}$ is shown as following:

$$m_{t,i} = [\pi_{t,i}, s_{t,i}, k_{t,i}], \quad 0 \leq i < \gamma, \;\; 0 \leq t < T \quad (3)$$

The state of path $m_{t,i}$ at time instant t is denoted as $n_{t,i}$, and the last state in $\pi_{t,i}$ is also denoted as $\pi_{t,i}^{last}$. The transition cost is set to -0.2. The path scores s, transition record k and sequence record π are updated with every new P_t as following:

$$s_{t+1,i} = s_{t,i} + p_{t+1,i} + \delta, \quad \delta = \begin{cases} -0.2, & \text{if } n_{t+1,i} \neq \pi_{t,i}^{last} \\ & \text{and } k_{t,i} < K \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$\pi_{t+1,i} = \begin{cases} \pi_{t,i} \cup n_{t+1,i}, & \text{if } n_{t+1,i} \neq \pi_{t,i}^{last} \text{ and } k_{t,i} < K \\ \pi_{t,i}, & \text{otherwise} \end{cases} \quad (5)$$

$$k_{t+1,i} = \begin{cases} k_{t,i} + 1, & \text{if } n_{t+1,i} \neq \pi_{t,i}^{last} \text{ and } k_{t,i} < K \\ k_{t,i}, & \text{otherwise} \end{cases} \quad (6)$$

In order to reduce computation, we limit the number of paths in *M* to γ, which is set to 300. The working mechanism of the proposed Viterbi-like decoder is given in algorithm 1. Fig. 6 depicts the illustration of our Viterbi-like decoder. Our decoder can inherently deal with the ambiguities at phoneme transitions as it naturally makes use of temporal ensembling.

Algorithm 1 Viterbi-like decoder for 3-tuple gesture recognition

function VITERBI-LIKE DECODER(P, S)
  Initialize s, π and k at P_0;
  for each P_t **do**
    Create all possible paths
    Update s, π and k according to (4), (5) and (6)
    Descending sort all m in M with their scores s
    Keep no more than the first γ paths
  end for
  return π of m with maximum s and k=K
end function

3.3. Training Details

In the trainings, we have used Stochastic Gradient Descent (SGD) with standard categorical cross-entropy loss. While we have used 5×10^{-4} and 1×10^{-3} weight decay for 2D and 3D CNNs, respectively, the momentum is kept same as 0.9 for all the trainings. As Jester is the largest available hand gesture dataset [7], we have pretrained all models on Jester dataset before fine tuning on SHGD-15 and SHGD-13. For 2D CNN models, before Jester pretraining, we also have used models pretrained with ImageNet as starting point. The learning rate for 2D CNNs is initialized at 0.001 and reduced with a factor of 0.1 at 25^{th}, 35^{th} and 45^{th} epochs. For trainings of 3D CNNs on Jester dataset, learning rate is initialized with 0.1 and reduced twice with a factor of 0.1 at 30^{th} and 45^{th} epochs. All trainings are completed at 60^{th} epoch for Jester and SHGD.

For fine tuning of SHGD-15 and SHGD-13, the pretrained parameters are loaded except for the first convolutional layer and the last fully connected layer. The number of input channels for the first convolutional layer is modified from 3 (RGB) to 2 for IR+D and 1 for IR and Depth modalities. In the last fully connected layer, the number of

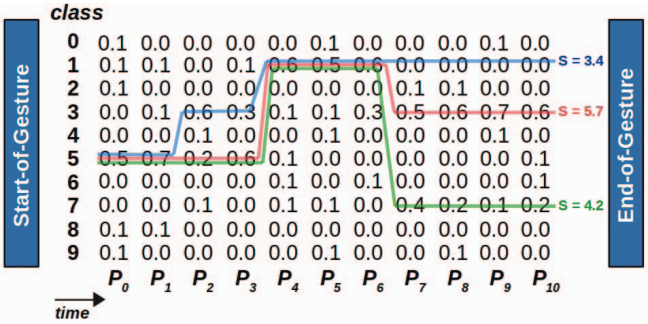


Figure 6: Illustration of our Viterbi-like decoder for 3-tuple gesture recognition. For the sake of simplicity, we have highlighted only three paths while the correct one is in red. For the correct path, π = [5,1,3], s = 6.1 and k = 2. 2 times the transition cost of 0.2 is subtracted from each path.

Model	Params	MFLOPs	Acc.(%)
2D-SqueezeNet	0.89M	310	87.40
2D-MobileNetV2	2.41M	366	91.35
3D-SqueezeNet	1.85M	686	87.74
3D-MobileNetV2	2.39M	344	93.33

Table 4: Results of different models on the validation set of Jester dataset. For 2D CNNs, FLOPs are calculated for extracting one frames features and final fc layers.

output features is set to the number of classes in SHGD. For SHGD-13, we have deployed weighted categorical cross-entropy loss as it is an unbalanced dataset.

We have deployed several data augmentation techniques such as random rotation ($\pm 10^{\circ}$), random resizing and random spatial cropping. Apart from spatial augmentations, we also applied temporal augmentations. Input clips are selected from random temporal positions given the bounds of each class. Moreover, at pretraining of 2D CNNs on Jester dataset, frames are selected randomly within each segment of videos as in Temporal Segment Network (TSN) [31], which introduces extra variation in the trainings.

4. Experiments

4.1. Results using Jester dataset

Jester is currently the largest available hand gesture dataset. There are in total 148.092 video samples collected for 27 different classes. As the labels of the test set are not publicly available, we have experimented on the validation set of the dataset. Table 4 summarizes the achieved results for our models. Besides the classification accuracy, the computational complexity in terms of floating point operations (FLOPs) and number of parameters are also given in Table 4 in order to highlight the resource efficiency of our models. The best result is achieved by 3D-MobileNetV2 with accuracy of 93.33%.

4.2. Results using SHGD-15 and SHGD-13

The performance of our models for SHGD-15 and SHGD-13 using different modalities are given in Table 5. The best results are achieved by 2D-SqueezeNet (98.47%) and 3D-MobileNetV2 (96.06%) for SHGD-15 and SHGD-13, respectively, both at IR+D modality.

For SHGD-15, 2D CNNs always achieve better results than 3D CNNs for all modalities. This is because of the fact that around 66.67% of samples in SHGD-15 are static gestures, and 2D CNNs captures static content better than 3D CNNs. On the other hand, around 20% of samples in SHGD-13 are static gestures resulting 3D CNNs to perform better. In order to highlight this situation, we have plotted the receiver operating characteristics (ROC) curves for static phoneme classes; and dynamic preparation and retraction classes in SHGD-13, which can be seen in Fig. 7, where the same results can be observed.

Model		Accuracy (%)	
		SHGD-15	SHGD-13
IR	2D-SqueezeNet	98.13	92.56
	2D-MobileNetV2	97.36	93.11
	3D-SqueezeNet	92.99	95.87
	3D-MobileNetV2	92.85	94.62
Depth	2D-SqueezeNet	98.13	95.02
	2D-MobileNetV2	98.13	95.64
	3D-SqueezeNet	89.93	95.87
	3D-MobileNetV2	92.78	95.85
IR+D	2D-SqueezeNet	98.47	93.94
	2D-MobileNetV2	97.92	95.06
	3D-SqueezeNet	92.64	95.59
	3D-MobileNetV2	94.31	96.06

Table 5: Results of different models with different modalities on the test sets of SHGD-15 and SHGD-13.

Different models are sensitive to different data modalities. For instance, 2D-MobileNetV2 performs better at depth modality, whereas 3D-MobileNetV2 performs best at IR+D modality. However, fusion of different modalities (IR+D) results in better performance most of the time.

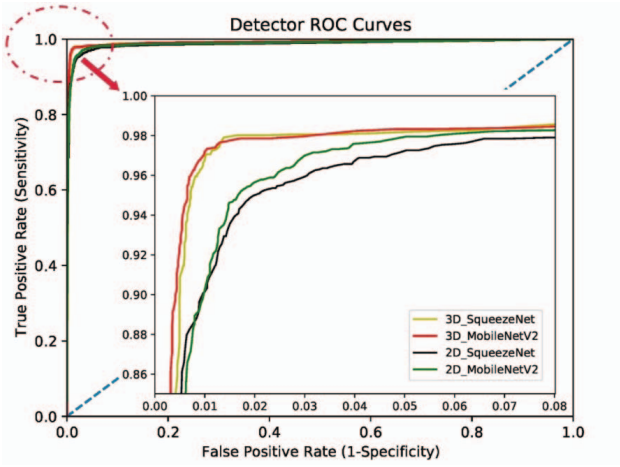
4.3. Results for 3-tuple gesture recognition

In this section, we evaluate the performance of our models for 3-tuple gesture recognition. Test set for this objective contains 1620 samples from 810 different permutations (i.e. classes). In order to evaluate the performance, three different errors and the total accuracy are defined as following:

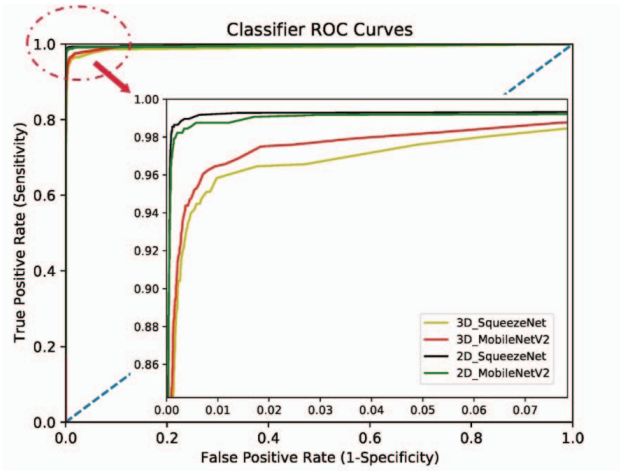
- Detector error: The number of the gesture tuples, in which SoG or EoG is not successfully detected. It includes the flags detected at the wrong time and flags not detected at all.
- Tuple error: The number of the gesture tuples, whose predicted sequence does not match to the ground truth.
- Single error: The number of the single phonemes which are recognized mistakenly inside the tuple error. For instance, if the ground truth is [6,8,10] and the recognized tuple is [6,10,12], then the single error is 2.
- Total accuracy: The percentage of the correctly predicted tuples in the whole test set, where $N_{samples}$ is equal to 1620. It is calculated as following:

$$Acc{=}(1 - \frac{Err_{det} + Err_{tup}}{N_{samples}})\% \qquad (7)$$

For this task, models are trained with SHGD-13. Table 6 gives the performance of experimented models on different



(a)



(b)

Figure 7: ROC curves of 4 different models trained on SHGD-13 with IR+D modality. **(a)** Average ROC curves for dynamic preparation and retraction classes, **(b)** Average ROC curves of all the static phoneme classes.

modalities for 3-tuple gesture recognition. For the detection threshold of detector, 5 and 6 are used for 2D and 3D CNNs, respectively. Similar to previous results, 3D CNNs capture dynamic classes better and make less detector errors. On the other hands, 2D CNNs make less tuple and single error as they consist of static classes.

3D-MobileNetV2 achieves the best performance with an accuracy of 94.69% for recognizing 810 different gesture tuples. 3D CNNs surpass 2D CNNs in this task generally, except for depth modality. We assume that this is due to the noise pixels appearing in depth modality from time to time. Therefore, 3D CNNs fail to capture the temporal relations between noisy frames.

	Model	Error			Acc.(%)
		Det	Tup	Sin	
IR	2D-SqueezeNet	191	54	126	84.88
	2D-MobileNetV2	116	103	248	86.60
	3D-SqueezeNet	11	159	375	89.51
	3D-MobileNetV2	10	209	492	86.48
Depth	2D-SqueezeNet	73	127	275	87.65
	2D-MobileNetV2	77	111	259	88.40
	3D-SqueezeNet	68	200	261	83.46
	3D-MobileNetV2	82	169	271	84.51
IR+D	2D-SqueezeNet	125	79	184	87.41
	2D-MobileNetV2	41	71	165	93.09
	3D-SqueezeNet	7	103	228	93.21
	3D-MobileNetV2	3	83	171	94.69

Table 6: Performance for the tuple detection. Det, Tup and Sin refer to the number of detector, tuple and single phoneme errors out of 1620 test samples.

5. Conclusion and Outlook

In this paper, we propose a novel approach for scaling hand gestures such that CNNs can recognize without requiring an enormous quantity of training data or extra training effort. For this objective, we create and share a benchmark dataset, Scaled Hand Gestures Dataset (SHGD), which contains gesture tuples having a sequence of gesture phonemes. Moreover, we have proposed a network architecture for recognition of gesture tuples using a novel Viterbi-like decoder. In our experiments, we have used the 2D and 3D versions of the SqueezeNet and MobileNetV2 models. We achieve a classification accuracy of 98.47% for 15 single gesture classes, and we achieve an accuracy of 94.69% for recognition of 810 different 3-tuple gesture classes.

The proposed approach contains utmost importance in order to meet the needs of applications requiring more complex HCI systems. We can easily scale hand gestures exponentially by increasing the number of gesture phonemes in multi-tuple gestures.

Similar to Rotokas language (spoken on the island of Bougainville), which contains 11 phonemes, we plan to create a hand language by using multi-tuple gestures and start talking with our hands.

Acknowledgements

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU, and Infineon Technologies with the donation of Pico Monstar ToF camera used for this research.

References

[1] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017.

[2] N. H. Dardas and N. D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement*, 60(11):3592–3607, 2011.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[5] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301, 1995.

[6] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

[7] T. B. N. GmbH. The 20bn-jester dataset v1. https://20bn.com/datasets/jester, 2019.

[8] P. M. X. Y. S. Gupta and K. K. S. T. J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. CVPR, 2016.

[9] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*, pages 18–22, 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[15] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. *arXiv preprint arXiv:1901.10323*, 2019.

[16] O. Köpüklü, N. Köse, and G. Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. *arXiv preprint arXiv:1804.07187*, 2018.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[19] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 5, 2018.

[20] Q. Miao, Y. Li, W. Ouyang, Z. Ma, X. Xu, W. Shi, and X. Cao. Multimodal gesture recognition based on the resc3d network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3047–3055, 2017.

[21] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[22] P. Molchanov, S. Gupta, K. Kim, and K. Pulli. Multi-sensor system for driver's hand-gesture recognition. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–8. IEEE, 2015.

[23] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016.

[24] J.-L. Nespoulous, P. Perron, and A. R. Lecours. *The biological foundations of gesture: Motor and semiotic aspects*. Psychology Press, 2014.

[25] L. Prasuhn, Y. Oyamada, Y. Mochizuki, and H. Ishikawa. A hog-based hand gesture recognition system on a mobile device. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 3973–3977. IEEE, 2014.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520. IEEE, 2018.

[27] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[28] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[29] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[30] J. Wan, Y. Zhao, S. Zhou, I. Guyon, S. Escalera, and S. Z. Li. Chalearn looking at people rgb-d isolated and continuous datasets for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 56–64, 2016.

[31] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856. IEEE, 2018.