A Review On
Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks

Richeek Das

May 25, 2020

1 Introduction

Here, I am going to summarize a paper by **Okan Kopuklu, Ahmet Gunduz, Neslihan Kose, Gerhard Rigoll from the Institute for Human-Machine Communication, TU Munich, Germany and Dependability Research Lab, Intel Labs Europe, Intel Deutschland GmbH, Germany**. Its a quite recent paper of 2019 titled "Real-time Hand Gesture Detection and Classification using Convolutional Neural Networks" and it aims to address issues faced by real-time recognition of dynamic hand gestures.

The above mentioned problem is difficult because there is no indication when a gesture starts and ends in a video feed and on top of that there are memory and power budget issues as well. In this paper, the authors address those challenges by proposing a hierarchical structure enabling offline-working convolutional neural network architecture to operate online efficiently by using sliding window approach.

They have evaluated their proposed architecture on two publicly available datasets - **EgoGesture** and **NVIDIA Dynamic Hand Gesture Datasets** - which require temporal detection and classification of the performed hand gestures. **ResNeXt-101** model, which is used as a classifier, achieves the *state-of-the-art* offline classification accuracy of **94.04%** and **83.82%** for *depth modality* on *EgoGesture* and *NVIDIA benchmarks*, respectively.

2 Methodology

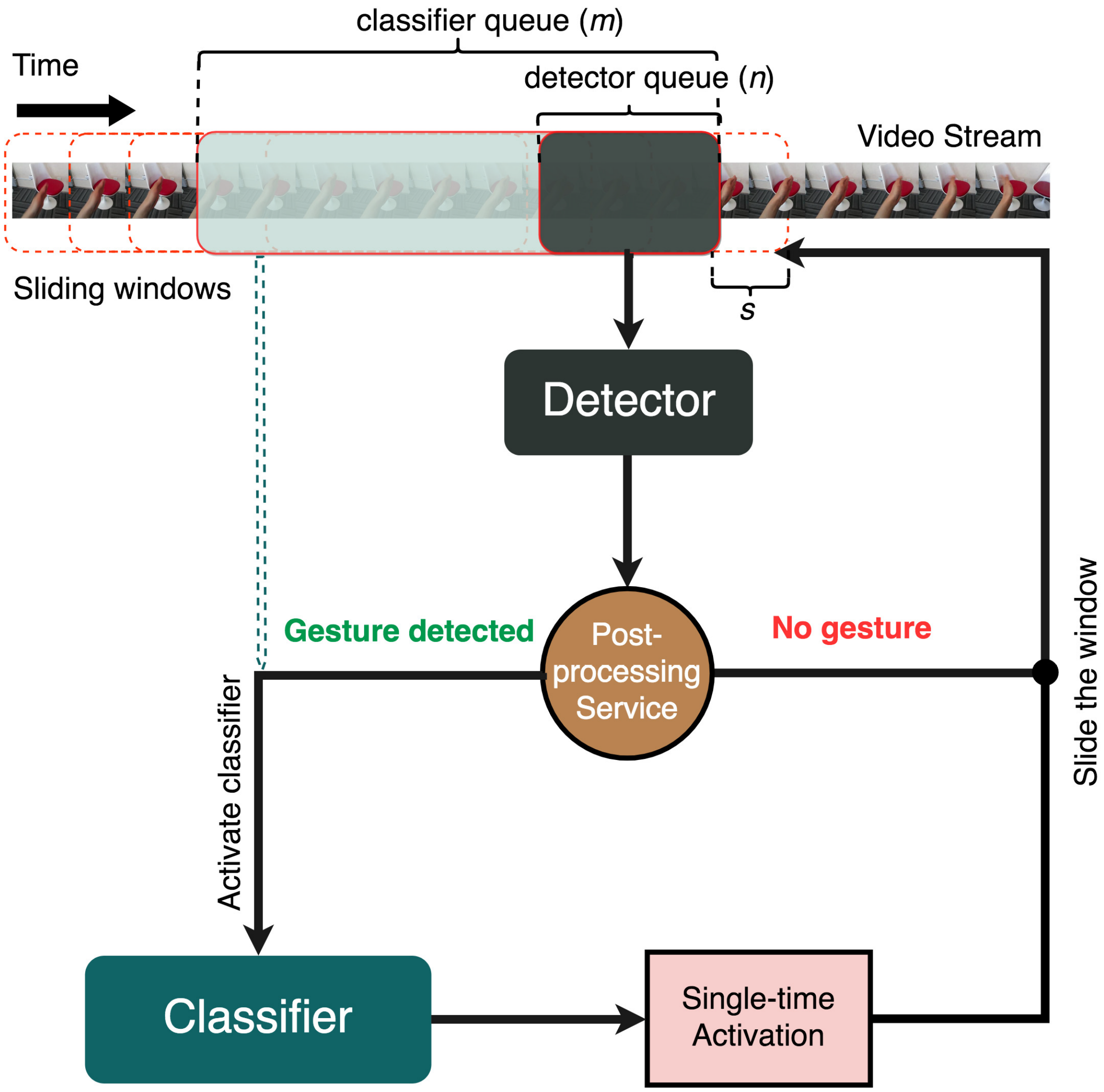
In real-time gesture recognition applications, there are several characteristics that the system needs to satisfy: (i) An acceptable classification accuracy, (ii) fast reaction time, (iii) resource efficiency and (iv) single-time activation per each performed gesture and this paper probably for the first time implements single-time activations on deep learning based hand gesture recognition.

They have taken a two-model hierarchical architecture and that enables the state-of-the-art CNN models to be used in real-time applications with high efficiency.

A detailed overview :

2.1 Architecture

The figure below depicts the workflow used for an efficient real-time recognition system using *sliding window approach*. The workflow starts with a detector which is used as a switch to activate classifier if a gesture gets detected. The detector and classifier models are fed by a sequence of frames with size n and m respectively, such as $n \ll m$. The stride value used for the sliding window is represented by s. Higher stride provides less resource usage. In addition to the *detector* and *classifier models*, *one post-processing* and *one single-time activation* service is introduced to the workflow.

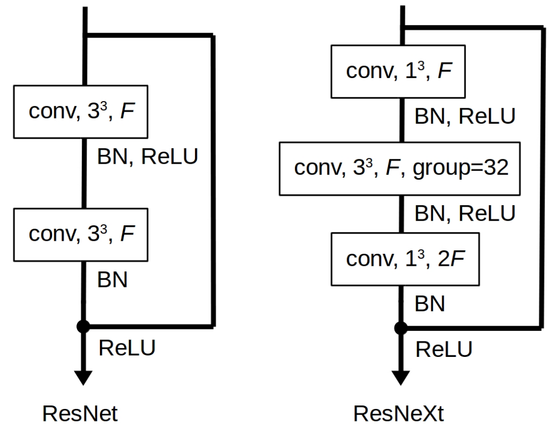


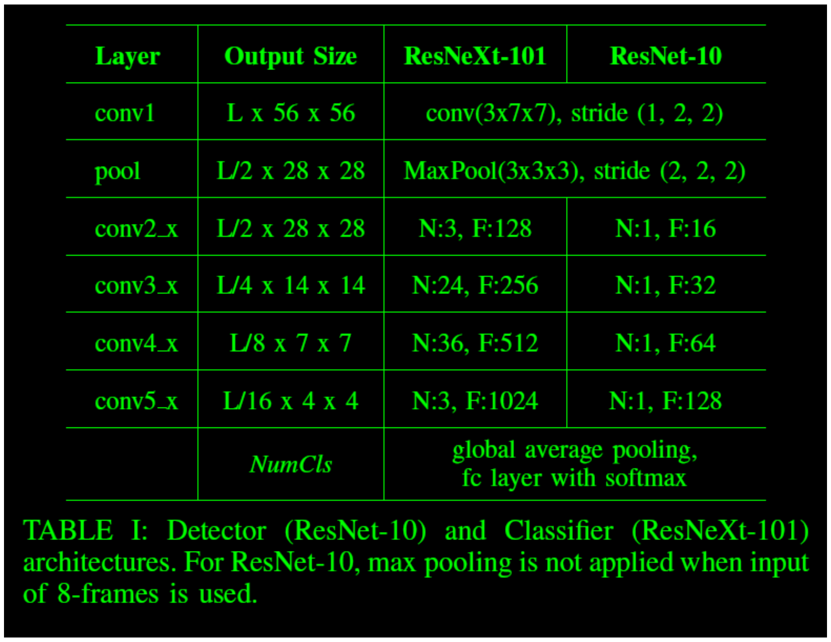
The general workflow of the proposed two-model hierarchical architecture. Sliding windows run through incoming video frames where detector queue placed at the very beginning of classifier queue. If the detector recognizes an action/gesture, then the classifier is activated. The detector's output is post-processed for a more robust performance, and the final decision is made using single-time activation block where only one activation occurs per performed gesture.

The blocks in detail :

1. **Detector :** The purpose of the detector is to distinguish between gesture and no gesture classes by running on a sequence of images, which detector queue masks. To improve the overall accuracy, the detector runs on smaller number of frames than the classifier and the detector queue is placed on the very beginning of classifier queue. Moreover, the detector model is trained with a weightedcross entropy loss in order to decrease the likelihood of false positives

2. **Classifier :** Since there is no limitation regarding the size or complexity of the model, any architecture providing a good classification performance can be selected as classifier. For *ResNeXt-101*, they have chosen the model parameters as given in **Table I** with ResNeXt block as given in the figure below.





Layer	Output Size	ResNeXt-101	ResNet-10
conv1	L x 56 x 56	conv(3x7x7), stride (1, 2, 2)	
pool	L/2 x 28 x 28	MaxPool(3x3x3), stride (2, 2, 2)	
conv2_x	L/2 x 28 x 28	N:3, F:128	N:1, F:16
conv3_x	L/4 x 14 x 14	N:24, F:256	N:1, F:32
conv4_x	L/8 x 7 x 7	N:36, F:512	N:1, F:64
conv5_x	L/16 x 4 x 4	N:3, F:1024	N:1, F:128
	NumCls	global average pooling, fc layer with softmax	

TABLE I: Detector (ResNet-10) and Classifier (ResNeXt-101) architectures. For ResNet-10, max pooling is not applied when input of 8-frames is used.

3. **Post-processing :** In dynamic hand gestures, it is possible that the hand gets out of the camera view while performing gestures. Even though the previous predictions of the detector are correct, any misclassification reduces the overall performance of the proposed architecture. In order to make use of previous predictions, we add the raw softmax probabilities of

the previous detector predictions into a queue (q_k) with size k, and apply filtering on these raw values and obtain final detector decisions. With this approach, detector increases its confidence in decision making.

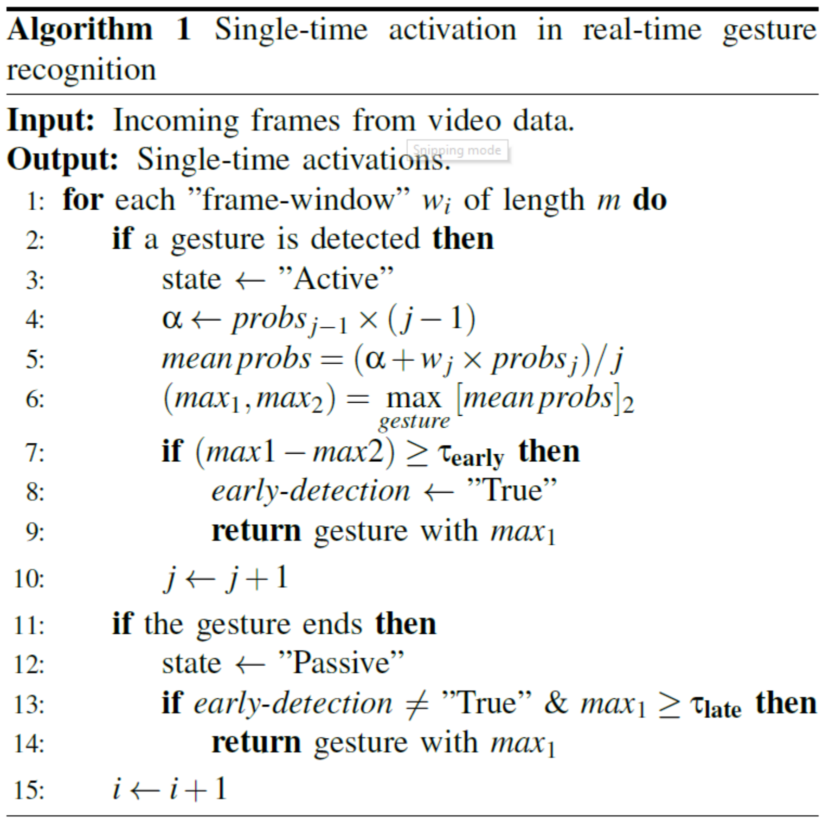
4. **Single-time Activation :** Dynamic gestures have preparation, nucleus and retraction parts. Nucleus is the most discriminative one, since we can decide which gesture is performed in nucleus part even before it ends. The most critical part of the early-detection is that, the gestures should be detected after their nucleus parts for a better recognition performance. Because several gestures can contain a similar preparation part which creates an ambiguity at the beginning of the gestures. To curb this error they have applied weighted-averaging on class scores with a weight function :

$$w_j = \frac{1}{(1 + exp^{-0.2 \times (j-t)})}$$

where, j is the iteration index of an active state at which a gesture is detected abd t is calculated by using :

$$t = \left\lfloor \frac{\mu}{4 \times s} \right\rfloor$$

where μ corresponds to the mean duration of the gestures in the dataset and s is the stride length. With this weighted-averaging strategy, they have forced the singletime activator to make decision at mid-late part of the gestures after capturing their nucleus parts.



Algorithm 1 Single-time activation in real-time gesture recognition

Input: Incoming frames from video data.
Output: Single-time activations.

```
1: for each "frame-window" $w_i$ of length $m$ do
2:     if a gesture is detected then
3:         state ← "Active"
4:         $\alpha \leftarrow probs_{j-1} \times (j-1)$
5:         $meanprobs = (\alpha + w_j \times probs_j)/j$
6:         $(max_1, max_2) = \max_{gesture} [meanprobs]_2$
7:         if $(max1 - max2) \geq \tau_{\textbf{early}}$ then
8:             early-detection ← "True"
9:             return gesture with $max_1$
10:        $j \leftarrow j+1$
11:    if the gesture ends then
12:        state ← "Passive"
13:        if early-detection ≠ "True" & $max_1 \geq \tau_{\textbf{late}}$ then
14:            return gesture with $max_1$
15:    $i \leftarrow i+1$
```

5. **Evaluation of the Activations :** Problems faced include (i) Misclassification of the gesture due to the classifier (ii) Not detecting the gesture due to the detector (iii) Multiple detections

in a single gesture. Considering these scenarios, they have proposed to use the **Levenshtein distance** as our evaluation metric for online experiments. The Levenshtein distance is a metric that measures distance between sequences by counting the number of item-level changes (insertion, deletion, or stitutions) to transform one sequence into the other.

3 Conclusion

This paper has presented to us a novel two-model hierarchical architecture for real-time hand gesture recognition systems. The proposed architecture provides resource efficiency, early detections and single time activations and are easily integrable to state-of-the-art models overtime.

The sliding window mechanism as mentioned here, surely looks very promising with the confidence values it reports and on top of that, it also seems to be highly scalable with the clearcut problem statements it proposes. So, that should be the key takeaway of this paper.

4 References

- The Original Paper
- Deep Learning with Python by **François Chollet**
- Towards Data Science