Hand Gesture Classification Using Grayscale Thermal Images and Convolution Neural Network

Rakesh Reddy Yakkati[1], Sreenivasa Reddy Yeduri[2], and Linga Reddy Cenkeramaddi[2]
[1]Department of Mathematics, Birla Institute of Technology, Mesra, Jharkhand 835215, India
[2]Department of ICT, University of Agder, Grimstad-4879, Norway
Email: yrakesh2109@gmail.com, yeduris@uia.no, linga.cenkeramaddi@uia.no

***Abstract*—In this paper, we propose a convolution neural network for classifying grayscale images of hand gestures. For classification, we look at ten different hand gestures collected from various people using a thermal camera. We then compare the proposed model's performance in terms of classification accuracy and inference time to that of other benchmark models. We demonstrate through extensive results that the proposed model achieves higher classification accuracy while using a smaller model size. Furthermore, we show that the proposed model outperforms benchmark models in terms of inference time.**

***Index Terms*—Convolution neural network, image classification, hand gesture, classification accuracy, and inference time.**

I. Introduction

Image classification has attracted the remote-sensing community due to its application in environmental and socioeconomic applications [1], [2]. It helps in taking a decision based on the outcome. The need for classification arises when we place the image of an object in a group or class based on the attributes of the object [3]–[5]. In order to improve the classification accuracy, the scientists and practitioners have made enormous efforts in developing novel classification models [6], [7].

In [8], the authors have considered binary Bayesian classifier to classify the images based on whether they captured in indoor or outdoor. Further, the the images captured in outdoor are classified based on city or landscape. Finally, the landscape images are classified into sunset, forest, and mountain [8]. A probabilistic model has been proposed in [9] for supervised topic models in order to jointly recognize, classify, and annotate the images. It has been concluded in [9] that the proposed model has resulted in improved classification accuracy in comparison to the state-of-the-art models. A novel between-class learning method has been proposed in [10] to classify the images from between-class. Here, between-class images have been obtained by mixing the images of two different classes in different ratios. The concept of mixing of images is motivated from mixing of sound signals. Then, the authors in [10] have considered a mixing method that considers each image as a waveform in order to improve the performance in terms of classification accuracy. In [11], the authors have proposed Fisher Vector representation as an alternative for Bag of Visual words encoding technique which is commonly used in image classification. In the proposed work, a set of low-level patch descriptors are extracted from each image as a first step. Thereafter, the images are classified based on the deviation of the extracting set from the universal generative model [11]. In [12], a feature mining paradigm has been proposed for image classification. Further, several feature mining strategies have been examined on the proposed paradigm for image classification. A Hierarchical Gaussianization (HG) model based image representation has been proposed in [13] that will incorporate both appearance and spatial information in the hierarchical structure. Further, a supervised dimension reduction technique has been proposed to enhance the discriminating power of HG representation. Finally, a nearest centroid classifier has been considered which has shown to be achieved higher classification accuracy [13]. In [14], the authors have surveyed the techniques to train the deep convolution neural network model in order to improve the accuracy. It has shown that the minor modifications made in model architecture, data preprocessing, loss function, and learning rate schedule has significantly improved the image classification accuracy of ResNet-50, Inception-V3, and MobileNet [14]. Motivated by this, in this work, we propose a convolution neural network for grayscale image classification of the hand gestures. The key contributions of this work are as follow:

- We consider ten different hand gestures collected from different people for classification.
- We then compare the performance of the proposed model with other benchmark models in terms of classification accuracy and inference time.
- Through extensive results, we show that the proposed model results higher classification accuracy with lower model size.
- Finally, we have implemented the proposed algorithm on the Raspberry Pi 4 Model B hardware.
- We show that the proposed model outperforms benchmark models in terms of inference time.

The complete data set and the algorithms can be accessed https://github.com/aveen-d/Radar$_{c}lassificationhere$ 1001[15].

The remainder of this paper is organized as follows. Section II provides the dataset details. The proposed CNN model is described in III. Section IV provides the numerical results in terms of classification accuracy and inference time. Finally, concluding remarks and possible future works are discussed

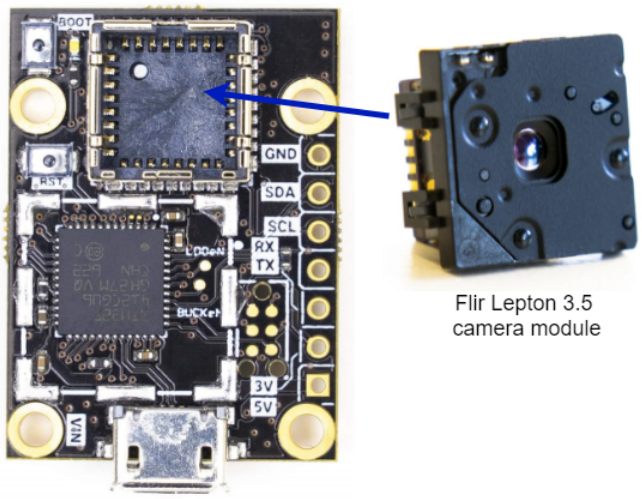

Fig. 1: The FILR LEPTON 3.5 camera module and Purethermal 2 breakout board.

TABLE I: Number of samples per class in the Dataset.

Class Name	Number of Images
a	360
b	360
c	360
d	360
e	360
f	360
g	360
h	360
i	360
j	360
Total	**3600**

in Section V.

II. Dataset Details

In this section, we describe the dataset in brief. We start with the hardware setup used for collecting the images and then we describe the details of the dataset.

A. Dataset Details

The dataset is created from the images captured from the FLIR Lepton 3.5 thermal camera. This thermal camera is a Long Wavelength Infra Red (LWIR) micro camera from FLIR. It has a 160×120 pixel resolution and a radiometric calibrated array of 19200 pixels [16]. The camera module is interfaced to the Purethermal 2 breakout board. It is a FLIR Lepton smart I/O module with pre-configured plug-and-play functionality via a USB port. The Purethermal 2 breakout board and FLIR LEPTON 3.5 camera module can be seen in Fig. 1. The camera is interfaced to the edge computing system, Raspberry pi model 4.

The dataset contains the 10 class grayscale image hand gestures (160×120 pixel size). Classes 1 through 10 are denoted by the letters 'a', 'b', $\cdots$, 'j'. With this, a total of

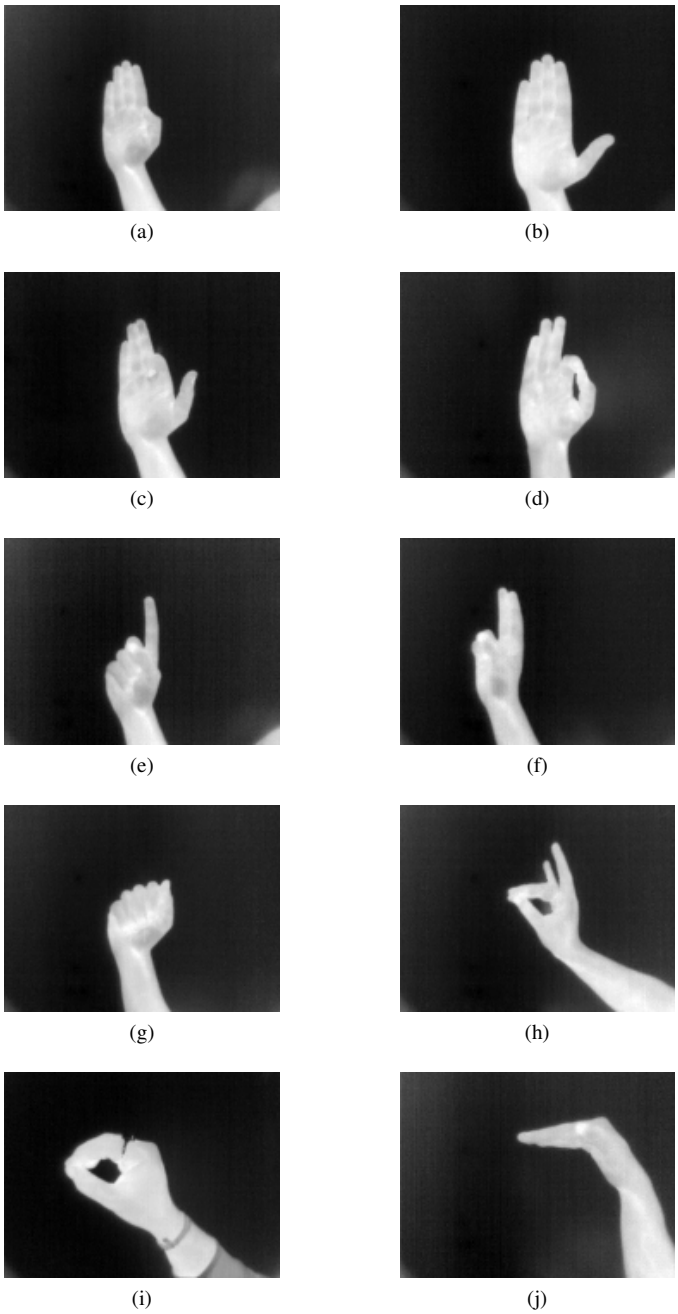

Fig. 2: A complete set of 10 different Gray hand Gestures.

3600 grayscale image hand gestures were created as shown Table I. Males and females between the ages of 20 and 60 are used to create grayscale hand gestures. Fig. 2 shows a complete set of 10 grayscale hand gestures of a person.

Usually color images have three channels and gray scale images are of single channel. Thus, the collected dataset contains gray scale images of single channel. However, all pre-trained models are designed to train with three channel images and cannot train with one channel. Thus, we have distributed the one dimensional image over three channels for a better fit.

Next, we describe the proposed CNN model architecture.

III. Proposed CNN model

The architecture of the proposed deep CNN model for the classification of gray scale hand postures is shown in Fig. 6. It consists of 10 layers such as input layer, Batch normalization layer, convolution 1 layer with tanh activation function, Max pooling 1 layer, convolution 2 layer with tanh activation function, Max pooling 2 layer, convolution 3 layer with tanh activation function, Max pooling 3 layer, flatten layer, dense layer with softmax activation function, and output layer. The original image of the dataset is 120×160 with 1 channel. However, the input image is converted into a gray scale image of 128×128 with 3 channel i.e, $128 \times 128 \times 3$ 3-dimensional matrix and the output layer size is 10 which represent the classes of gray hand gestures as shown in Fig. 2. The role of the convolution layer is to learn image features from the input later. It applies the filter on input layer to extract the feature maps in CNN model and the pooling layer is used to reduce the dimensionality of the input layer and helps the CNN to train faster. When the input layer passes through the convolution layer or pooling layer, the size of the output layer is calculated by

$$\text{Output Size} = \frac{n - f + 2p}{s} + 1\,, \tag{1}$$

where, n represents the input size, f represents the filter or kernel size or pooling size, p denotes the padding size, and s is stride size. The output of a convolutional layer of the consecutive feature map value can be computed as

$$V(x, y) = (i * k)[x, y] = \sum_a \sum_b k[a, b] i[x - a, y - b]\,, \tag{2}$$

where, 'i ' represents the input image, k denotes the kennel size of $(a * b)$ matrix, x and y represent the indexes of the output matrix.

The max pooling layer and average pooling layer are the most commonly used pooling layers. The output of the max pooling layer can be obtained as

$$M(i, j) = max(I(i, j)) \tag{3}$$

In CNN classification model, the final layer is a fully connected layer or dense layer with softmax activation function, collecting the information for final feature maps. The output of the dense layer and activation function softmax is computed by

$$F_i = \frac{1}{1 + e^{-(w_i x_{i-1} + b_i)}} \tag{4}$$

The output of the softmax layer is computed as

$$a_i = \frac{e_i^a}{\sum_{n=1}^{N} e_n^a}\,, \tag{5}$$

where, W_i represents the weights and b_i represents the bias in the dense layer. In softmax, N denotes the number of output classes. In this work, the cross-entropy loss for multi class classification is calculated as

TABLE II: CNN architecture

Model: "sequential"		
Layer (type)	Output Shape	Param #
batch_normalization (BatchNo	(None, 128, 128, 3)	12
conv2d (Conv2D)	(None, 128, 128, 8)	1952
max_pooling2d (MaxPooling2D)	(None, 42, 42, 8)	0
conv2d_1 (Conv2D)	(None, 42, 42, 16)	10384
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 16)	0
conv2d_2 (Conv2D)	(None, 14, 14, 32)	41504
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 32)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 10)	5130
Total params: 58,982		
Trainable params: 58,976		
Non-trainable params: 6		

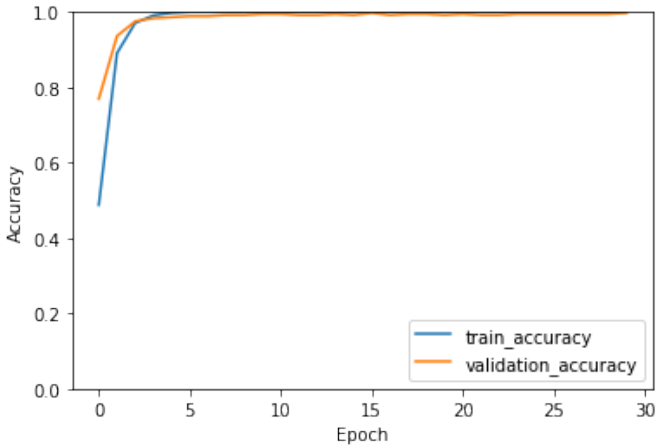

Fig. 3: The training and validation accuracy of the proposed deep CNN model.

$$\text{Cross entropy loss} = \frac{-1}{\sum_{i=1}^{k} (y_i \log(g_i) + (1 - y_i) \log(1 - y_i))}\,, \tag{6}$$

where, k is the number of classes, y_i is the actual output of the CNN, g_i is the predicted probability of the output class. In this work, we have used a total 3600 samples which makes 360 samples for each class. With a partition rate of 0.1, these samples are divided for training and testing i.e., out of 3600 samples 90% (3240 samples) are used for training and 10% (360 samples) are used for validation and testing. We have applied adam solver algorithm which is a combination of MSprop and Stochastic Gradient Descent with momentum. Adam solver algorithm used to update weights and bias parameters in convolution layer and dense layer.

IV. Numerical Results

In this section, we present the results corresponding to the accuracy, loss, and confusion matrix of the proposed model. We then provide the comparison results in terms of accuracy

TABLE III: Performance comparison of the proposed model with other benchmark models in terms of validation accuracy, total number of parameters, number of trainable and non-trainable parameters, and size of the model.

Model	Validation Accuracy (10-fold)	validation Accuracy	Total Parameters	Trainable Parameters	Non-trainable Parameters	Size
OurModel	99.52 (+- 0.37)	99.72	58,982	58,976	6	744 KB
DenseNet121	99.41(+-0.40)	99.44	7,048,779	11,275	7,037,504	27.7 MB
DenseNet169	99.69(+-0.23)	99.72	12,659,530	16,650	12,642,880	49.4 MB
DenseNet201	99.61(+-0.37)	100	18,343,115	21,131	18,321,984	71.3 MB
InceptionResNetV2	98.80(+-0.63)	99.16	54,352,106	15,370	54,336,736	208.6 MB
InceptionV3	99.13(+-0.43)	99.16	21,825,323	22,539	21,802,784	83.9 MB
MobileNetV2	99.66(+-0.27)	100	2,272,075	14,091	2,257,984	9.1 MB
MobileNetV3Large	99.38(+- 0.36)	99.16	4,239,242	12,810	4,226,432	16.6 MB
MobileNetV3Small	99.16 (+- 0.62)	99.16	1,540,218	10,250	1,529,968	6.2 MB
NASNetMobile	98.61(+-0.755)	97.77	4,280,286	10,570	4,269,716	17.6 MB
ResNet50V2	99.30(+-0.28)	99.72	23,585,290	20,490	23,564,800	90.4 MB
ResNet101V2	99.36(+-0.17)	99.16	42,649,099	22,539	42,626,560	163.5 MB
ResNet152V2	99.17(+-0.41)	98.88	58,352,138	20,490	58,331,648	223.6 MB
VGG16	99.72 (+- 0.27)	99.44	14,719,818	5,130	14,714,688	56.2 MB
VGG19	99.55 (+- 0.46)	100	20,029,514	5,130	20,024,384	76.5 MB
Xception	99.16(+-0.56)	99.44	20,884,019	22,539	20,861,480	80.1 MB
EfficientNetB0	99.66 (+- 0.24)	99.72	4,062,381	12,810	4,049,571	16 MB
EfficientNetB1	99.69 (+- 0.31)	100	6,588,049	12,810	6,575,239	25.8 MB
EfficientNetB2	99.47(+- 0.47)	98.89	7,782,659	14,090	7,768,569	30.3 MB
EfficientNetB3	99.55 (+- 0.28)	99.72	10,798,905	15,370	10,783,535	41.9 MB
EfficientNetB4	99.3 (+- 0.50)	99.16	17,691,753	17,930	17,673,823	68.3 MB
EfficientNetB5	99.44 (+- 0.48)	99.44	28,534,017	20,490	28,513,527	109.9 MB
EfficientNetB6	99.25(+- 0.44)	98.88	40,983,193	23,050	40,960,143	157.5 MB
EfficientNetB7	99.33 (+- 0.39)	99.72	64,123,297	25,610	64,097,687	246 MB

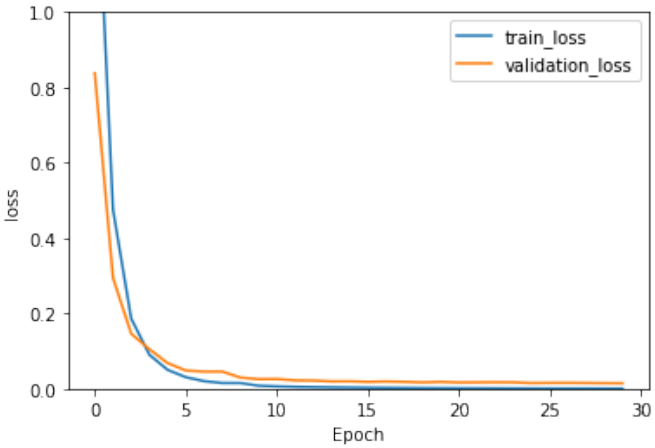

Fig. 4: The training and validation loss of the proposed deep CNN model.

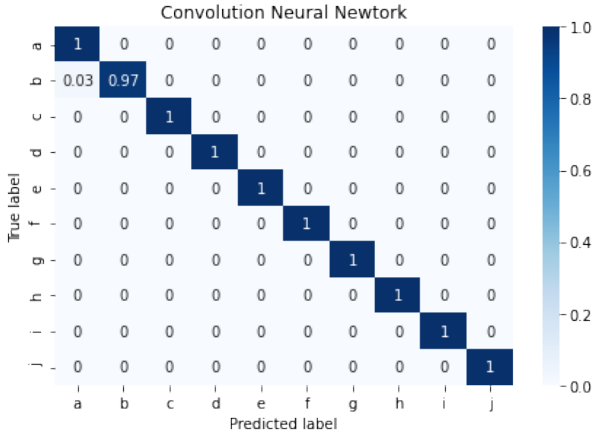

Fig. 5: Confusion matrix of the proposed deep CNN.

and inference time to validate the performance of the the proposed model and other benchmark models.

The summary in terms of output shape and number of parameters for the proposed model is listed in Table II. The variation of the accuracy and loss for the proposed model for training and validation of the dataset is shown in Figs. 3 and 4. From Fig. 3, it is observed that the accuracy increases exponentially with epoch and saturates at a maximum of 99.72% after 10th epoch. From Fig. 4, it is observed that the value of loss decreases exponentially with epoch and saturates at a minimum at 13th epoch. The confusion matrix of the proposed CNN model on the dataset is shown in Fig. 5. It is observed from Fig. 5 that the proposed model classifies the classes “a”, “c”, “d”, “e”, “f”, “g”, “h”, “i”, and “j” with an accuracy of 100% of their respective class’s test dataset. However, the images corresponding to class “b” can be identified with an accuracy of 97% of their respective class’s test dataset.

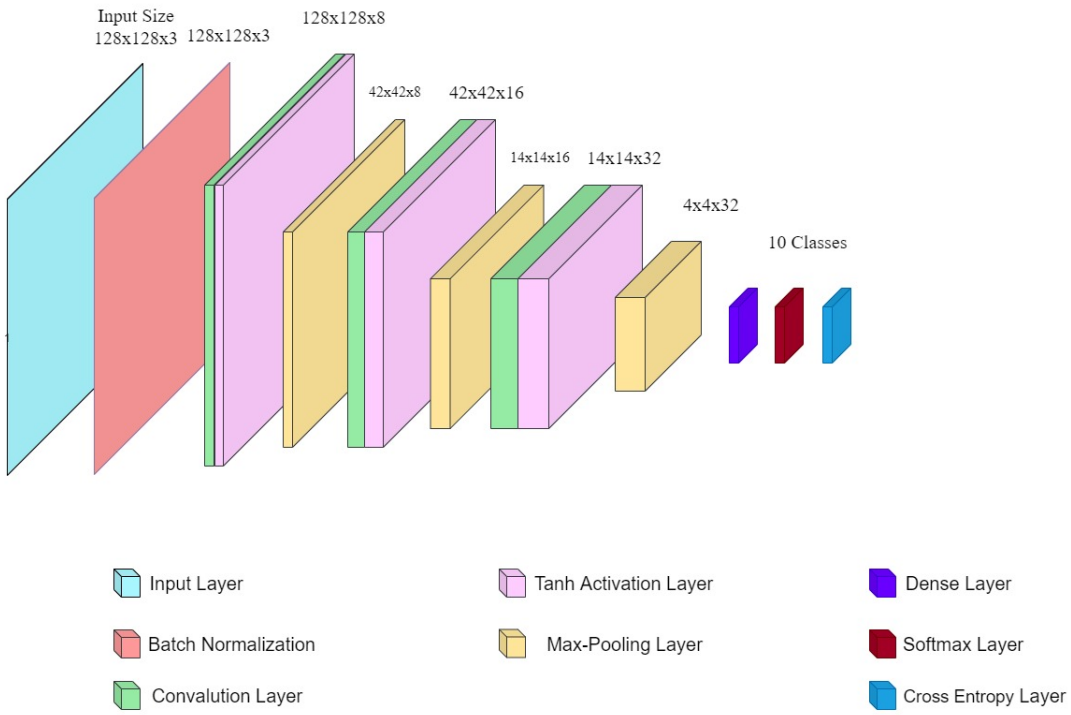

Fig. 6: The architecture of the proposed CNN model.

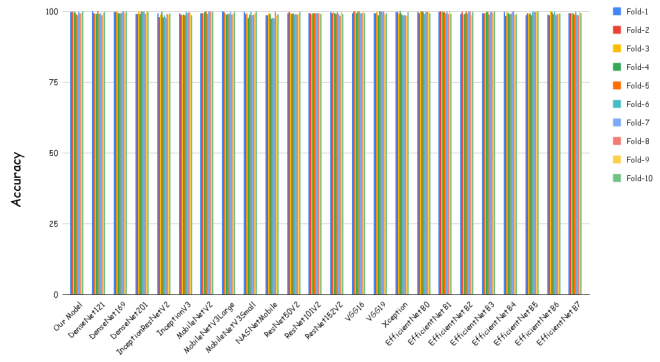

Fig. 7: Performance comparison of the proposed model with Pre-trained models in terms of achievable accuracy in each fold.

In addition to the proposed model, we also trained and validated some of the pre-trained models such as MobileNet, VGG,ResNet, DenseNet, Xception, Inception, NASNetMobile, and EfficientNet with the considered dataset. We have evaluated the 10-fold validation and holdout validation with 10% split as shown in Table III. The accuracy values at each fold for the proposed model and 23 pre-trained models is shown in Fig 7. We compare the proposed model with the considered pre-trained models in terms of accuracy, size, and number of parameters of the dataset as given in Table III. From Table. III, it is observed that the proposed model and considered pre-trained models result in the similar classification accuracy in the range of 97%-100%. Further it is observed that even though considered pre-trained models achieve comparable classification accuracy of the proposed model, the number of parameters and size of the proposed model is very less as compared to other benchmark models. The pre-trained models which preform better in terms of classification accuracy has large model size in range of 6MB – 224MB. However, the proposed model size is 774 KB which is comparably very less. Thus, we conclude that the proposed model delivers better classification accuracy with a model size of less than 1MB.

The performance comparison of the proposed model to other pre-trained models in terms of inference time on different GPUs such as Tesla T4, Tesla P4, Tesla K80, and GTX 1050, CPUs like Intel i5 8th gen and Intel(R) Xeon(R) Platinum, and hardware like Raspberry Pi 4 and AGX Jetson Xavier is listed in Table IV. It can be observed from Table IV that the inference time is very less for the proposed model as compared to other pre-trained models.It is also noticed that the proposed model takes 0.2 milliseconds (ms) inference using GPU Tesla T4. From Table IV, it can be observed that the proposed model inference time is in the range of 0.2 ms to 0.75 ms on GPUs. While using raspberry Pi 4 and AGX Jetson xavier the time taken for proposed model is 26.41 ms and 39.04 ms, respectively.

V. Conclusion

In this paper, we have proposed a convolution neural network for classification of grayscale images of the hand gestures. We considered ten different hand gestures collected from different people for classification. We compared the

TABLE IV: Inference time comparison of the proposed model with other benchmark models.

2***Models**	**Inference Time**							
	Tesla T4	**Tesla P4**	**Tesla K80**	**NVIDIA GeForce GTX 1050**	**Intel core i5 8th gen**	**Intel(R) Xeon(R) Platinum 8175M CPU @ 2.50GHz**	**Raspberry pi 4**	**AGX**
Our Model	**0.2**	**0.58**	**0.71**	**0.74**	**2.68**	**9.11**	**26.41**	**39.04**
DenseNet121	3.52	6.51	9.46	10.31	65.62	116.78	565.7	226.26
DenseNet169	4.14	7.19	12.81	12.36	83.34	142.84	592.47	204.77
DenseNet201	5.53	8.69	14.42	15.16	119.32	183.93	739.44	219.18
InceptionResNetV2	6.93	8.38	16.25	-	-	142.99	762.17	181.1
InceptionV3	3.03	4.84	7.88	8.32	39.64	66.62	306	113.92
MobileNetV2	1.74	2.72	4.04	5.32	24.16	34.92	122.63	70.19
MobileNetV3Large	1.59	2.42	4.85	3.77	21.98	27.35	121.53	82.26
MobileNetV3Small	0.51	1.41	2.03	2.3	9.57	11.05	56.71	71.37
NASNetMobile	3.02	4.9	7.38	8.55	40.88	67.56	202.83	145.58
ResNet152V2	7.71	15.49	22.78	21.35	148.88	248.69	1291.68	161.48
ResNet101V2	6.5	7.81	13.51	14.96	107.6	172.11	865.79	135.69
ResNet50V2	3.15	4.6	7.84	9.23	58.03	92.92	453.89	103.38
VGG16	5.91	7.65	11	14.91	126.49	307.66	1471.72	1398.78
VGG19	6.05	8.99	14.62	16.94	164.51	378.87	1734.32	1584.31
Xception	6.39	7.15	14.47	12.76	75.2	135.47	524.72	117.72
EfficientNetB0	1.82	3.95	4.35	5.64	44.16	61.29	213.29	88.26
EfficientNetB1	2.71	4.52	7.38	7.51	54.8	81.62	292.7	110.78
EfficientNetB2	2.78	6.39	7.27	7.9	58.99	85.78	320.35	114.7
EfficientNetB3	3.41	7.31	8.16	10.18	78.63	121.54	428.4	128.84
EfficientNetB4	6.37	8.06	12.33	13.39	106.97	158.87	607.59	151.12
EfficientNetB5	6.55	12.66	15.59	18.34	131.55	221.79	862.88	180.56
EfficientNetB6	10.55	14.6	24.77	23.5	172.3	295.85	1161.29	196.3
EfficientNetB7	13.62	23.99	27.9	31.74	236.41	399.51	1655.14	239.97

proposed model's performance to that of other benchmark models. We demonstrated through extensive results that the proposed model resulted in higher classification accuracy with a smaller model size. Furthermore, when deployed on different CPUs and GPUs, the proposed model outperforms benchmark models in terms of inference time.

ACKNOWLEDGMENT

This work was supported by the Indo-Norwegian Collaboration in Autonomous Cyber-Physical Systems (INCAPS) project: 287918 of the INTPART program and the Low-Altitude UAV Communication and Tracking (LUCAT) project: 280835 of the IKTPLUSS program from the Research Council of Norway.

REFERENCES

[1] P. Aplin, P. Atkinson, and P. Curran, "Per-field classification of land use using the forthcoming very fine spatial resolution satellite sensors: problems and potential solutions," 1999.

[2] J. Stuckens, P. Coppin, and M. Bauer, "Integrating contextual information with per-pixel classification for improved land cover classification," *Remote Sensing of Environment*, vol. 71, no. 3, pp. 282–296, Mar. 2000.

[3] S. S. Nath, G. Mishra, J. Kar, S. Chakraborty, and N. Dey, "A survey of image classification methods and techniques," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014, pp. 554–557.

[4] P. GONG and P. J. HOWARTH, "Land-use classification of spot hrv data using a cover-frequency method," *International Journal of Remote Sensing*, vol. 13, no. 8, pp. 1459–1471, 1992. [Online]. Available: https://doi.org/10.1080/01431169208904202

[5] M. Pal and P. M. Mather, "An assessment of the effectiveness of decision tree methods for land cover classification," *Remote Sensing of Environment*, vol. 86, no. 4, pp. 554–565, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425703001329

[6] F. J. Gallego, "Remote sensing and land cover area estimation," *International Journal of Remote Sensing*, vol. 25, no. 15, pp. 3019–3047, 2004. [Online]. Available: https://doi.org/10.1080/01431160310001619607

[7] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, 2007. [Online]. Available: https://doi.org/10.1080/01431160600746456

[8] A. Vailaya, M. Figueiredo, A. Jain, and H.-J. Zhang, "Image classification for content-based indexing," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 117–130, 2001.

[9] W. Chong, D. Blei, and F.-F. Li, "Simultaneous image classification and annotation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1903–1910.

[10] Y. Tokozume, Y. Ushiku, and T. Harada, "Between-class learning for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[11] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[12] P. Dollar, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[13] X. Zhou, N. Cui, Z. Li, F. Liang, and T. S. Huang, "Hierarchical gaussianization for image classification," in *2009 IEEE 12th International Conference on Computer Vision*, 2009, pp. 1971–1977.

[14] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] A. Dayal, *Radar_classification*, (accessed April, 2020). [Online]. Available: https://github.com/aveen-d/Radar_classification

[16] FLIR Systems. Flir lepton 3 & 3.5. [Online]. Available: https://lepton.flir.com/wp-content/uploads/2015/06/Lepton3_3.5-Data-Sheet.pdf