Neural Computing and Applications (2023) 35:12029–12037
https://doi.org/10.1007/s00521-023-08337-y

ORIGINAL ARTICLE

DL-DARE: Deep learning-based different activity recognition for the human–robot interaction environment

Sachin Kansal[1] · Sagar Jha[1] · Prathamesh Samal[1]

Received: 27 June 2022 / Accepted: 25 January 2023 / Published online: 18 February 2023
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2023

Abstract

This paper proposes a deep learning-based activity recognition for the Human–Robot Interaction environment. The observations of the object state are acquired from the vision sensor in the real-time scenario. The activity recognition system examined in this paper comprises activities labeled as classes (pour, rotate, drop objects, and open bottles). The image processing unit processes the images and predicts the activity performed by the robot using deep learning methods so that the robot will do the actions (sub-actions) according to the predicted activity.

Keywords RGB sensor · VGG-16 · Resnet-50 · Hyper-parameters · Deep learning · Visual tracking · Feature extraction

1 Introduction

Deep neural network (DNN) implementations of deep learning (DL) approaches have been widely accepted because of their high-staging processing power. With unstructured data, deep learning can process a wide range of features, giving it enormous power and reliability. The issue of object posture prediction in real-time tracking has become a significant concern. Real-time tracking is now possible because of sophisticated sensors, intelligent chips and control theory, much like in the 1990s. As a result of the soaring image capture rate, the activity recognition framework's vision systems continued to struggle with posture estimation. Robots must do the actuation in that amount of time. Parallel robots have a larger load-carrying capacity than serial robots and are extensively employed for grabbing goods. To create a deep learning-based model, several different algorithms were put out. This research proposes to implement a deep learning framework for activity recognition. The primary focus of this research is on developing an activity recognition system based on deep learning. Repositioning an object skillfully has been defined as manipulation [1]. These systems have a wide range of uses, ranging from heavy industry to smart homes (Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12).

A combination of Random Forests and Hidden Markov Models (HMM) performs the best, according to the work of Roitberg et al. [2], who developed various machine learning algorithms utilizing leave-one-out cross-validation. The construction of feature vectors that are appropriate for activity recognition and a comparison of several machine learning methods for feature importance estimation and classification make up their two-step methodology. Recent computer vision research has largely employed data from 2D videos to focus on the identification of human activities [3–5]. The use of 3D skeleton data for activity detection has grown in popularity over the past few years [6, 7], due to Shotton et al. [8] development of a reliable real-time approach for skeleton capturing with random forests.

On 2D + X data volumes, the article found two issues with ML (machine learning). 2D picture observation is X, and 2D is a variable associated with depth, wavelength,

✉ Sachin Kansal
sachin.kansal@thapar.edu

[1] Computer Science Engineering Department, Thapar Institute of Engineering Technology Patiala, Patiala, Punjab 147004, India

time, etc. [9]. DLA-based medical image recognition, classification, and segmentation are discussed in this work. Medical image analysis using DLA is made easier with the help of this guide [10]. It introduces the concept of sparse representation into the architecture of deep learning networks in this paper (Multilayer nonlinear mapping is reported to complete the complicated function approximation in deep learning [11]. Research in this paper examines how to use a convolutional neural network (CNN) to classify pneumonia based on a chest X-ray dataset [12].

This paper inputs pre-therapy lung CT images into Deep Profiler. This multi-task deep neural network incorporates radionics into the training process to generate an image fingerprint that predicts time-to-event treatment outcomes and approximates the classical technique's radiomic features [13]. This paper defines image-based deep learning to predict complexity as the need for component separation and pulmonary and wound complications after Abdominal Wall Reconstruction (AWR) reported in [14].

In [15], the researchers have proposed a technique that employs action proposals to first extract and categorize useful motion characteristics using a ConvNet framework and then use action proposals to identify one human action in videos, independent of camera movement. Convolution-basedmodels are very powerful for solving image and video recognition problems; hence, we can use such models for our task of predicting the activity from a given video sequence.

Algorithm Predict the activity recognition from the given MIME dataset using the deep learning approaches. The real-time estimation and prediction of the activities are performed specifically to the activity predicted by the robot.

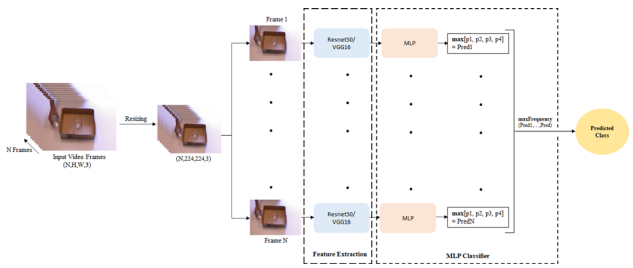

Fig. 1 Activity classification framework

Algorithm Predict the activity recognition from the given MIME dataset using the deep learning approaches. The real-time estimation and prediction of the activities are performed specifically to the activity predicted by the robot.

Input: Captured image (RGB)
Output: *Predict the activity recognition from the given MIME dataset using the deep learning approaches.*
Begin
While (true)
1. Compute the pose of the object(s) involved in the scene.

2. Model Training (Can be iterated for n batches)
For i in batch_size:
For video, label in video_dataset:
N←num_frames(video)
for frame_no in range(0,N,skip_by):
temp_frame ← extract_frame(frame_no,video)
temp_frame ← temp_frame.reshape(224,224,3)
video_frames ← append(temp_frame)
video_labels ←append(label)
class_probabilites ← model(video_frames)
classification_loss ← loss_fn(class_probabilities,video_labels)

3. Performing the Deep Learning (DL) based pose prediction after the model training.

N ← num_frames(test_video)
for frame_no in range(N):
temp_frame ← extract_Frame(frame_no,test_video)
temp_frame ← temp_frame.reshape(224,224,3)
class_probability ← append(model(temp_frame))

class_prediction ← maxFrequency(class_probability)

3. Robots perform the task predicted by the proposed framework.
End while
Begin
The offline training of the model using the Deep learning methods is done.
End

2 Methodology

2.1 Proposed method

Videos contain very rich semantic information. Inspired by the huge success of the deep learning methods in analyzing the image, audio, and text data, significant efforts are recently being devoted to the design of deep nets for video

Fig. 2 Activity classification images extracted from the training video (class: POUR)

analytics. Video classification serves as a fundamental and essential step in the process of analyzing video content. Instead of processing images initially, here, we have to classify videos into four tasks given above.

We can notice two features in videos, i.e., the temporal and spatial aspects, also known as spatial-temporal features. The temporal motion is converted to successive frames so that the conventional CNN and related architectures designed for images can be directly deployed. The detailed technique has been explained below:

(i) Each video contains multiple frames in time; hence, we initially extract all the frames. Once the frames are extracted, their name is saved along with a corresponding tag in the CSV file. This file helps to read the frames while processing and training them.
(ii) The next step is to create training and validation sets. In order to generate the validation set, it must be ensured that the distribution of each class in both the training and validation sets is similar. To accomplish so, we may employ the stratify parameter of the sci-kit-learn package, which maintains a consistent distribution of classes.
(iii) Due to the large dataset, a custom CNN model made from scratch may not work. So, pre-trained models have been used to define the architecture of the model. For our research, we used VGG16 and ResNet-50 for learning rich representations from the frames.
(iv) For the training and validation frames, features are extracted from the pre-trained models. We find the shape of the frames to be changed from (224, 224, 3) to (7, 7, 512) for each frame after passing it through the pre-trained network.
(v) For the final predictions, a Multi-Layered Perceptron (MLP) is used, which is a fully connected class of artificial neural network. It takes input in a single dimension. Hence, the features are reshaped resulting in a size of 25,088. It is to be also mentioned that the pixel values are normalized between 0 and 1 to aid the model for faster convergence.
(vi) Multiple FC layers are used along with dropout layers to prevent overfitting. The number of neurons in the final layer is equal to the number of classes to be predicted. In our case, it is four.

The model is now trained using the training frames. The optimum model is selected based on the validation loss.

2.2 Video pre-processing

Videos is a collection of frames, we check the video fps and then, treat each frame as an image and save it in the corresponding class folder. We skip some frames to reduce the time complexity. The spatial resolution of the frames is also converted into the shape (224, 224, 3) for uniformity.

2.3 Train-validation-split

We explored the dataset and created training and validation splits. We use a training set to train the model and a validation set to evaluate the trained model.

2.4 Model usage

We have used the Transfer Learning technique. This technique helps achieve the results faster and more accurately as the architecture of the models is already proven upon several previous tasks. The base architectures are downloaded from the TensorFlow hub. We must modify the end layers according to our problem statement of four classes.

3 Deep learning techniques

This section implements deep learning-based activity recognition for the Human–Robot Interaction environment.

3.1 Background

This section discusses the background of various deep learning techniques to make the vision-based pose prediction. Deep learning methods perform better than simple ANN, even though the training time of deep structures is higher than ANN. However, training time can be reduced using transfer learning GPU computing methods.

3.2 CNN

This section examines various deep learning techniques to predict the object's shape, and accordingly, catching can be performed based on the frames captured by the calibrated vision sensor. Convolutional Neural Networks (CNN) assign weights and biases to various objects in the image and differentiates one from another. It requires less pre-processing than other classification algorithms [16, 17]. CNN uses relevant filters to capture an image's spatial and temporal dependencies [18, 19]. The CNN architectures include LeNet, GoogleNet, AlexNet, VG-GNet, and ResNet.

Consider a real-time case of various activity recognition and implementation of the Deep Learning-based algorithm. Largest and most diverse ever demonstration dataset. It comprises 8260 human–robot demonstrations. There are 04 classes, namely, '**Pour', 'Rotate', 'Drop objects', and 'Open bottle'**. The sequence of activity classification images was extracted from the training video (class: POUR).

3.3 ResNet-50

ResNet-50 is a convolutional neural network that is 50 layers deep. It is a subclass of convolutional neural networks, with ResNet most popularly used for image classification. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database [20–23]. The pre-trained network can classify images into 1000 object categories. As a result, the network has learned rich feature representations for various images. The network has an image input size of 224-by-224. (Table 1)

3.4 VGG16

It is a Convolutional Neural Network (CNN) model proposed by Karen Simonyan and Andrew Zisserman at the University of Oxford. First and foremost, compared to the large receptive fields in the first convolutional layer, this model proposed the use of a very small 3 × 3 receptive field (filters) throughout the entire network with the stride of 1 pixel.

3.5 Testing methodology

We will take each video from the test set, extract frames, and save them in a temporary folder. At each iteration, we shall delete all other files from this folder. Next, we will read all of the frames from the temporary folder, use the pre-trained model to extract features from these frames, predict tags, and then, use the mode to assign a tag to that specific video and append it to the result. The model is then evaluated based on predicted and actual tags. We have used the accuracy score as the performance metric. It is to be noted that the training and testing videos are different, i.e., the testing videos are entirely new to the model. In short, we are passing a video for testing, converting the video into frames, assigning each frame a label, and then taking a

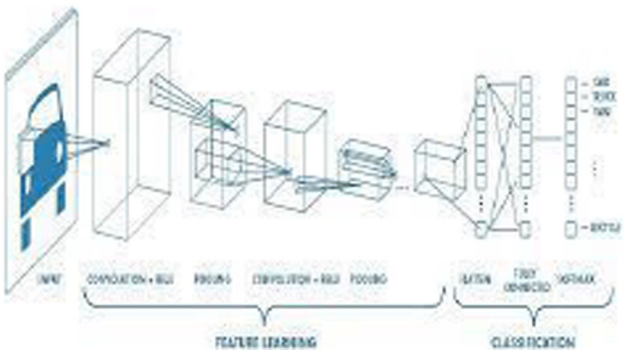

Fig. 3 Convolutional neural networks [28]

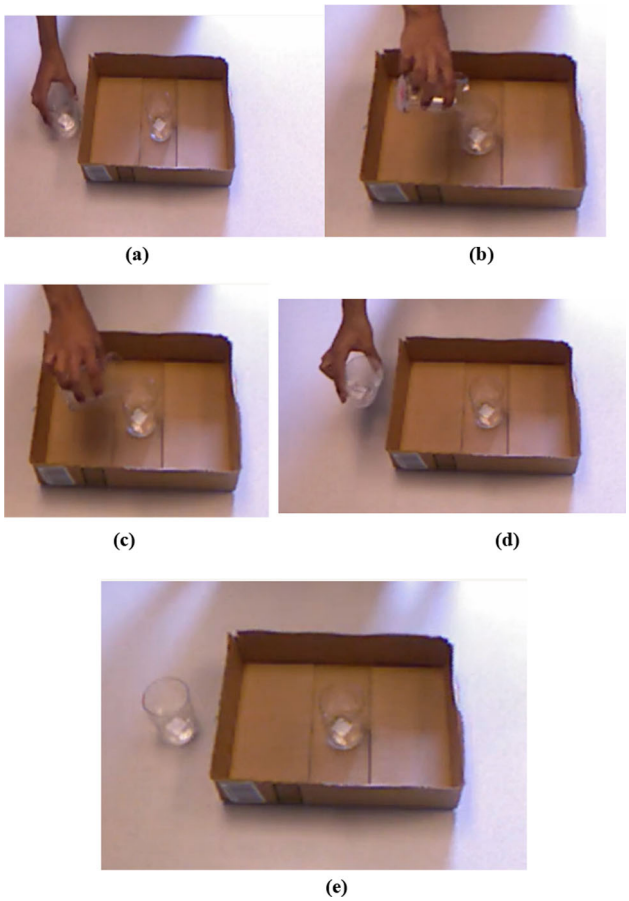

Fig. 4 (**a**–**e**). The sequence of activity classification images was extracted from the training video (class: POUR)

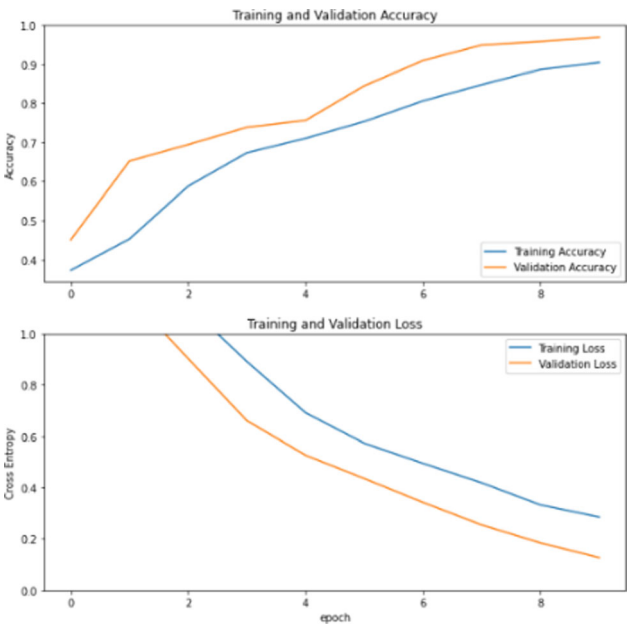

Fig. 5 ResNet50 (ADAM,0.0001)

mode to give a single label to the entire video since the task was of video classification.

3.6 GPU-enabled performance evaluation

In this section, GPU-enabled deep learning techniques using the Mixed Precision with Apex and Monitoring with Wandb are implemented to extract the appropriate state observations from the grabbed frames by the pre-calibrated camera. In this approach, torches are assigned to the GPU rather than copied from the CPU, i.e., We can reduce the time using the first approach. Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a model argument whose value is set before the learning process begins. The optimal hyperparameters control the underfitting and overfitting of the model. The key to machine learning algorithms is hyperparameter tuning, as shown in Table 2.

(a) Taken the training and testing dataset.
(b) Tuning the hyperparameters.
(c) Data splitting into training and testing.
(d) Apply data transform that includes augmentations and processing.
(e) Doing the custom dataset and Dataloader for the catching images.
(f) Implementing the optimizers:
(i) Stochastic Gradient Descent bs = 1; '*n*' number of examples. '*n* / 1' number of data loader/steps for 1 Epoch.
(ii) Mini-Batch Gradient Descent bs = 32; '*n*number of examples. '*n* /32' number of data loaders/step for 1 Epoch.
(iii) Full Batch Gradient Descent bs = total_number_of_samples number of data loader/steps = 1 for 1 Epoch.
(g) Loading the model.
(h) Computing the CrossEntropyLos (cel).
cel = Softmax(final activation function for normalizing the output of the FC Layer) + Negative Log-Likelihood (NLL) Loss.
(i) Train the model.
(j) Saving the model.

3.7 Various optimization parameters tuning and its performance evaluation

In this section, various optimization parameters tuning approaches are used with the GPU-enabled deep learning techniques using the Mixed Precision with Apex and Monitoring with Wandb are implemented to extract the appropriate state observations from the grabbed frames by the pre-calibrated camera. In this approach, torches are assigned to the GPU rather than copied from the CPU, i.e., We can reduce the time using the first approach. Hyperparameter tuning is choosing a set of optimal hyperparameters for a learning algorithm. Mixed precision uses 32-bit and 16-bit floating-point types in a model during training to make it run faster and use less

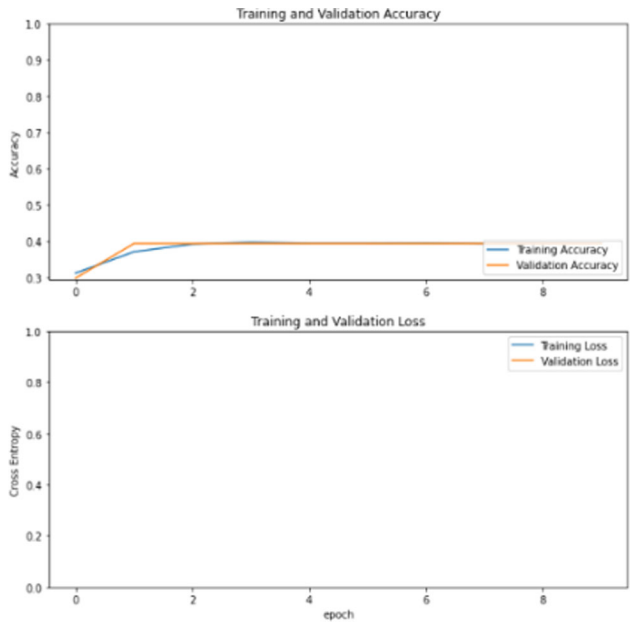

Fig. 6 ResNet50 (ADAM,0.1)

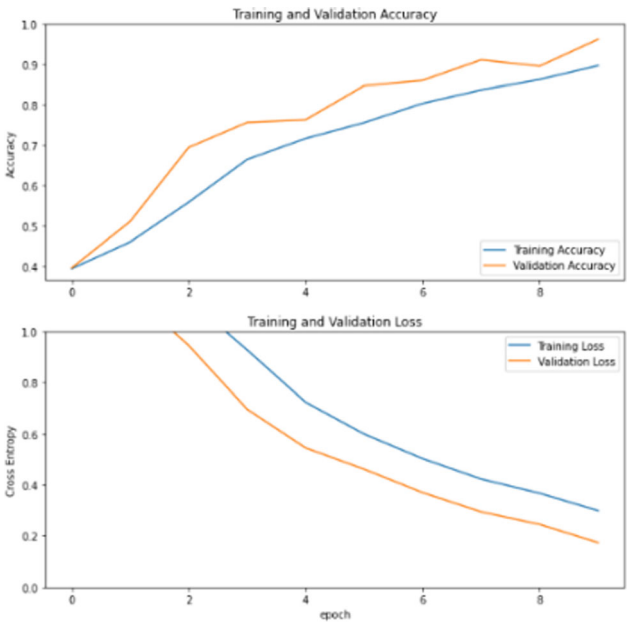

Fig. 7 ResNet50 (ADAM,0.001)

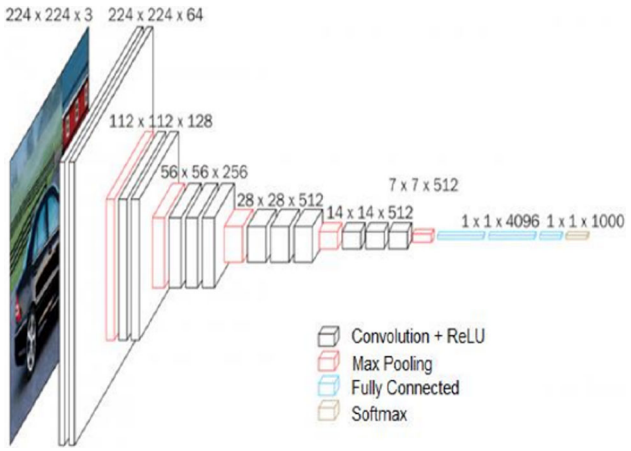

Fig. 8 VGG-16 Architecture

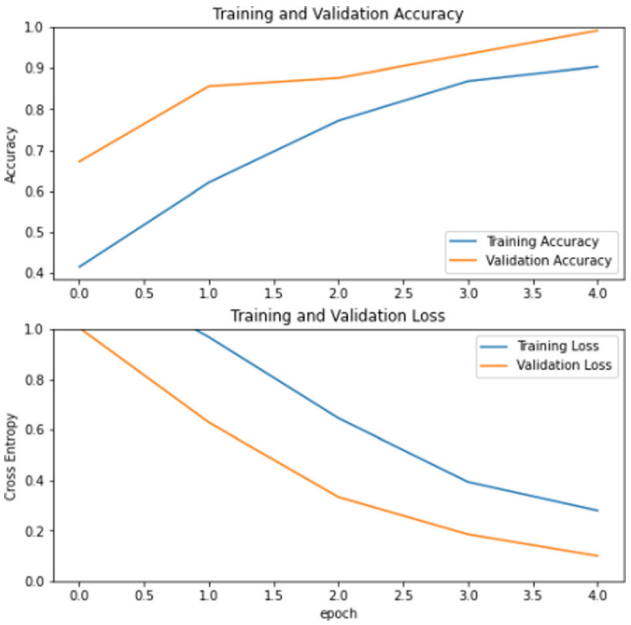

Fig. 9 VGG16 (ADAM,0.0001)

memory. Using mixed-precision training requires three steps:

1. To convert the model to the float16 data type where possible.
2. Keeping float32 master weights to accumulate per-iteration weight updates.
3. Using loss scaling to preserve small gradient values. Frameworks that support fully automated mixed-precision training also support:

(a) Automatic loss scaling and master weights integrated into optimizer classes,
(b) The automatic casting between float16 and float32 maximizes speed while ensuring no loss in task-specific

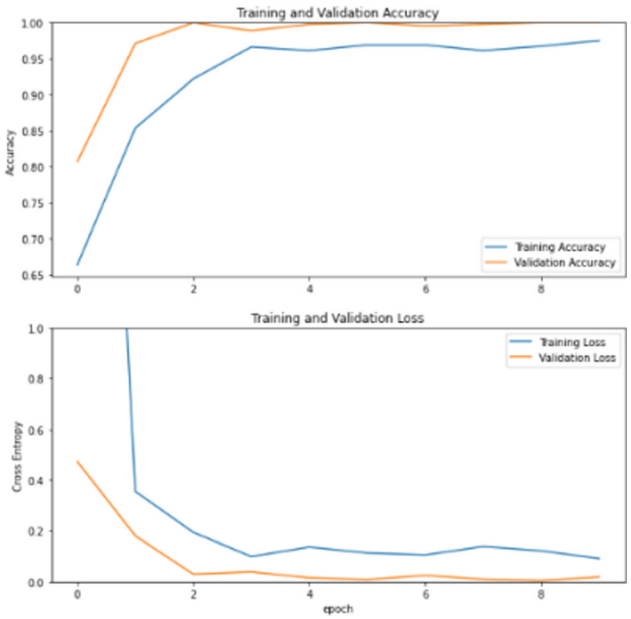

Fig. 10 VGG16 (ADAM,0.01)

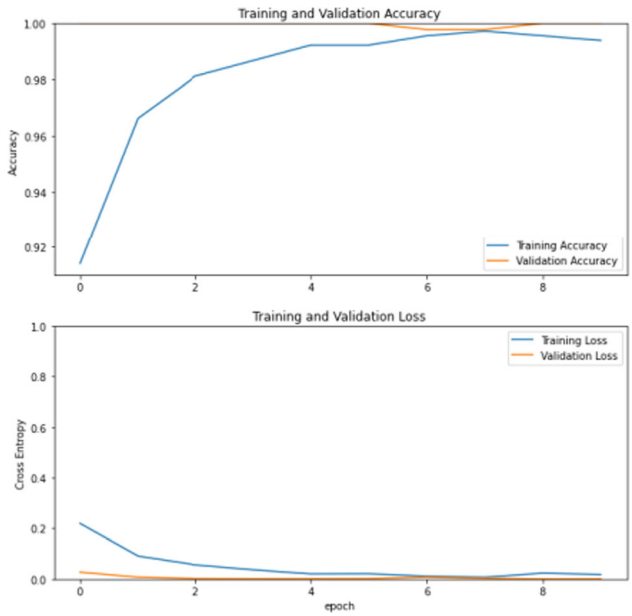

Fig. 11 VGG16 (ADAM,0.001)

accuracy. The key to machine learning algorithms is hyperparameter tuning, as shown in Table 3.

(a) Taken the training and testing dataset.
(b) Tuning the hyperparameters with performance tuning.
(c) Data splitting into training and testing.
(d) Apply data transform that includes augmentations and processing.

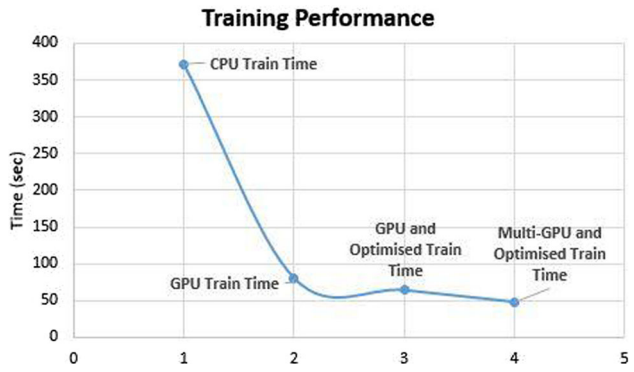

Fig. 12 Training performance

(e) Doing the custom dataset and Dataloader for the catching images.
(f) Create tensors directly on the target device.
(g) Enabling the TF32 on Ampere GPU.
(h) Enable channels_last memory format for computer vision models. This format is meant to be used with Automatic Mixed Precision (AMP) to further accelerate convolutional neural networks with Tensor Cores.
(i) Apex for Fused Optimizer and the AMP is used.
(j) Train the model.
(k) Saving the model

Table 1 Tuning Hyperparameters for performance tuning

Parameters	Value
Learning Rate	0.001
Epochs	10
BATCH_SIZE	8
IMAGE_SIZE	224
TRAIN_VALID_SPLIT	0.2
SEED	42
pin_memory	True
num_workers	0
channels_last	False
USE_AMP	True

Table 2 Tuning Hyperparameters

Parameters	Value
Learning Rate	0.001
Epochs	10
BATCH_SIZE	32
IMAGE_SIZE	224
TRAIN_VALID_SPLIT	0.2

Table 3 Tuning Hyperparameters for performance tuning

Parameters	Value
Learning Rate	0.001
Epochs	10
BATCH_SIZE	8
IMAGE_SIZE	224
TRAIN_VALID_SPLIT	0.2
SEED	42
pin_memory	True
num_workers	0
channels_last	False
USE_AMP	True

3.8 Multi-GPU with optimization parameters tuning enabled performance evaluation

In this section, various optimization parameters tuning approaches are used with the Multi-GPU-enabled deep learning techniques using the Mixed Precision with Apex are implemented to extract the appropriate state observations from the grabbed frames by the pre-calibrated camera. The key to machine learning algorithms is hyperparameter tuning, as shown in Table 4.

(a) Taken the training and testing dataset.
(b) Tuning the hyperparameters with performance tuning.
(c) Data splitting into training and testing.
(d) Apply data transform that includes augmentations and processing.
(e) Doing the custom dataset and Dataloader for the catching images.

Table 4 Tuning Hyperparameters for performance tuning

Parameters	Value
Learning Rate	0.001
Epochs	10
BATCH_SIZE	8
IMAGE_SIZE	224
TRAIN_VALID_SPLIT	0.2
SEED	42
pin_memory	True
num_workers	0
channels_last	False
USE_AMP	True
Distributed	True
World_size	2

(f) Create tensors directly on the target device.
(g) For distributed systems, after amp. initialize() function, wrap the model with apex.parallel.DistributedDataParallel() function.
(h) Enable channels_last memory format for computer vision models. This format is meant to be used with Automatic Mixed Precision (AMP) to further accelerate convolutional neural networks with Tensor Cores.
(i) Apex for Fused Optimizer and the AMP is used.
(j) Train the model.
(k) Saving the model.

4 Results

5 Conclusion

This paper uses deep learning-based modeling of the activity recognition system examined in this paper comprises activities labeled as classes (pour, rotate, drop objects, and open bottle). This paper also discussed the performance of the deep learning implementation in various architectures like CPU, GPU, optimized GPU, and multi-GPU optimization for the activity recognition framework. This research can be applied in the automation industry to track and manipulate goods while packaging.

Acknowledgements We hereby acknowledge the support of the Computer Science Engineering Department, Thapar Institute of Engineering Technology, Patiala, Punjab, for providing the facility.

Data availability The datasets analyzed during the current study are available in the [MIME Dataset] repository: [https://sites.google.com/view/mimedataset/dataset?authuser=0].

Declarations

Conflict of interest The authors do not have any conflict of interest.

References

1. David A, Chapman K, Weigelt M, Weiss D, Wel R (2012) Cognition, action and object manipulation. Psycholl Bull 138(5):924–946
2. Roitberg A, Perzylo A, Somani N, Giuliani M, Rickert M, Knoll A (2014) Human activity recognition in the context of industrial human-robot interaction, signal and information processing association annual summit and conference (APSIPA). Asia-Pacific 2014:1–10. https://doi.org/10.1109/APSIPA.2014.7041588
3. Poppe R (2010) A survey on vision-based human action recognition. Image Vis Comput 28(6):976–990
4. Turaga P, Chellappa R, Subrahmanian VS, Udrea O (2008) Machine recognition of human activities: A survey. IEEE Trans Circuits Syst Video Technol 18(11):1473–1488
5. Niebles JC, Wang H, Fei-Fei L (2008) Unsupervised learning of human action categories using spatial-temporal words. Int J Comput Vis 79(3):299–318
6. Ofli F, Chaudhry R, Kurillo G, Vidal R, Bajcsy R (2014) The sequence of the most informative joints: a new representation for human skeletal action recognition. J Vis Commun Image Represent 25(1):24–38
7. Papadopoulos GT, Axenopoulos A, Daras P (2014) Real-time skeleton-tracking-based human action recognition using kinect data. MultiMedia modeling. Springer, Berlin, pp 473–483
8. Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M, Moore R (2013) Real-time human pose recognition in parts from single-depth images. Commun ACM 56(1):116–124
9. Mahamane A, Benoit A, Lambert P (2020) Timed-image-based deep learning for action recognition in video sequences. Pattern Recognit 104:107353
10. Mualikrishna P, Ravi S (2021) Medical image analysis based on deep learning approach. Multimed Tools Appl 80:24365–24398
11. Liu JE, An FP (2020) Image classification algorithm based on deep learning-kernel function. Sci Program 2020:1–14
12. Samir Y, Shivajirao J (2019) Deep convolutional neural network-based medical image classification for disease diagnosis. J Big Data 6(1):1–18
13. Lou B, Doken S, Wingerter T, Gidwani M, Mistry N, Ladic L, Kamen A, Abazeed M (2019) An image-based deep learning framework for individualizing radiotherapy dose: a retrospective analysis of outcome prediction. Lancet Digit Health 1(3):e136–e147
14. Adib S, Eva B, Sullivan A (2021) Development and validation of image-based deep learning models to predict surgical complexity and complications in abdominal wall reconstruction. JAMA Surg 156:933–940
15. Rezazadegan F, Shirazi S, Upcrofit B, Milford M (2017) Action recognition: from static datasets to moving robots. IEEE Int Conf Robot Autom (ICRA) 2017:3185–3191. https://doi.org/10.1109/ICRA.2017.7989361
16. Mathew A, Amudha P, Sivakumar S (2021) Deep learning techniques: an overview. In: Hassanien A, Bhatnagar R, Darwish A (eds) Advanced machine learning technologies and applications. Springer, Singapore
17. Mathew A, Amudha P, Sivakumar S (2021) Deep learning models for medical Imaging In: Biomedical imaging devices and systems
18. Le QV et al (2015) A tutorial on deep learning part 2: autoencoders, convolutional neural networks and recurrent neural networks. Google Brain 20:1–20
19. Yamashita R, Nishio M, Do RKG, Togashi K (2018) Convolutional neural networks: an overview and application in radiology. Insights Imaging 9(4):611–629. https://doi.org/10.1007/s13244-018-0639-9
20. ImageNet. http://www.image-net.org. Accessed 28 May 2022
21. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. Archives Cornell University, New York
22. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
23. https://keras.io/api/applications/resnet/#resnet50-function. Accessed 28 May 2022

24. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed 28 May 2022
25. Kao ST, Ho MT (2021) Ball-catching system using image processing and an omni-directional wheeled mobile robot. MDPI Sens J 21(9):3208

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.