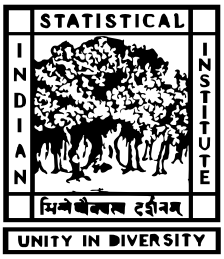

Kushal Singanporia, B.Tech. Electrical Engineering

Recognition of Strokes in Tennis Videos Using Deep Learning

Master's Thesis

to achieve the university degree of

Master of Technology

Master's degree programme: Computer Science

submitted to

Indian Statistical Institute

Supervisor

Dr. Kumar Sankar Ray

Indian Statistical Institute

Kolkata, July 2019

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Certificate

This is to certify that the dissertation titled **"Recognition of Strokes in Tennis Videos Using Deep Learning"** submitted by **Kushal Singanporia** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a *bona fide* record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Date	Signature

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.

Date	Signature

Acknowledgement

I am extremely grateful to my advisor Dr. Kumar Sankar Ray for constant support and encouragement. He provided me with exciting problems to think upon and was always available for discussions.

Abstract

Prior introduction of neural nets to domain of computer vision, action recognition requires specific domain knowledge. Still domain knowledge is useful in action recognition but with availability of huge data and neural nets, data-driven feature learning methods have emerged as an alternative. Recent trends in action recognition uses LSTM and its various modifications, as LSTM have memory retaining capability which other architectures lake. In this work we performed action recognition on different tennis strokes. Our work relay on architecture proposed By Husain, Dellen, and Torras, 2016. Architecture is comprised of various modified VGG-nets connected in parallel. As it doesn't include LSTM, which makes it different than other works.

Contents

List of Figures

1 Introduction

With increase in video contents now days, due to availability of cameras and storing facilities in abundance at comparatively low cost. We have lots of videos and it still increases. With popularization of video sharing websites we have tons of videos of different categories available to us. Also recent years we seen lots of improvement in deep learning domain. This improvement is still on increase with help of GPU advancement and availability of such hardware through cloud services at cheap rates.

Before introduction of deep learning methods to task of image/action recognition and classification, optical flow and descriptor based algorithms were vital tools to computer vision community. Yearly competition of image recognition attract attention of lots of researchers and deep learning enthusiast. Winners of this competition contribute lots of remarkable and accurate models like VGG, Alexnet etc. Success of this networks in tasks related to images inspire researchers to try them for videos in particular for action recognition. CNN and LSTM are in use for video action recognition and they also get pretty good accuracy.

One major benefit of using deep learning methods is it don't require domain expertise and knowledge, even without it if you have large enough good quality videos then you can train your neural network and eventually it will be able to identify and/or classify actions with decent accuracy.

1.1 What is action recognition

In action recognition a video or image is given and after processing on that we have to output a label which tells us about the action within that video or image.

1.1.1 Applications

Action recognition is useful in user-interfaces, automatic video organization or tagging, surveillance footage, search by video, gesture base interactive control, intelligent assisted living, home monitoring, crowd behaviour, event analysis etc.

1.1.2 Difficulties

- **Camera Movement**
  In case of hand-held camera, operator can cause it to shake. If camera is mounted on something and that support is in motion.
- **Occlusions**
  Action may not be fully visible if some other object is in foreground and our object is not completely visible.
- **Variance in Scale**
  People may appear at different scale in different videos, yet perform the same action.
- **Clutter Background**
  Other objects and/or humans presented in video frame.
- **Variation in Action**
  Different people perform actions in different ways.
- **Human Variation**
  Humans are of different sizes and shapes.
- **Etc.**

1.2 What is Deep Learning

Deep Learning is extension of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key

technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

1.2.1 Limitations with Deep Learning

While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

- Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.
- Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

1.2.2 Applications of Deep Learning

Automated driving, aerospace and defense, medical research, industrial automation, electronics, etc.

1.2.3 Deep learning as a tool for action recognition

Recently, deep learning methods such as convolutional neural networks and recurrent neural networks have shown capable and even achieve state-of-the-art results by automatically learning features from the raw sensor data.

As stated in the paper by Wang et al., 2019:

- The feature extraction and model building procedures are often performed simultaneously in the deep learning models. The features can be learned automatically through the network instead of being manually designed. Besides, the deep neural network can also extract high-level representation in deep layer, which makes it more suitable for complex activity recognition tasks.
- RNN and LSTM are recommended to recognize short activities that have natural order while CNN is better at inferring long term repetitive activities. The reason is that RNN could make use of the time-order relationship between sensor readings, and CNN is more capable of learning deep features contained in recursive patterns.

Neural networks has capability to automatically learn the features required to make accurate predictions from the raw data directly. This would allow new problems, new datasets, and new sensor modalities to be adopted quickly and cheaply.

2 Configuration

In this chapter we will discuss about VGG and then about our network which comprises of several modified VGG nets.

2.1 VGG16

VGG16 is a convolutional neural network model proposed by Simonyan and Zisserman, 2014 from the University of Oxford. It makes the improvement over AlexNet by Krizhevsky, Sutskever, and Hinton, 2012 mainly by replacing large kernel-sized filters with multiple small kernel-sized filters one after another.
The input to cov1 layer is of fixed size 224×224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.
Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000 way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all

networks.
All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

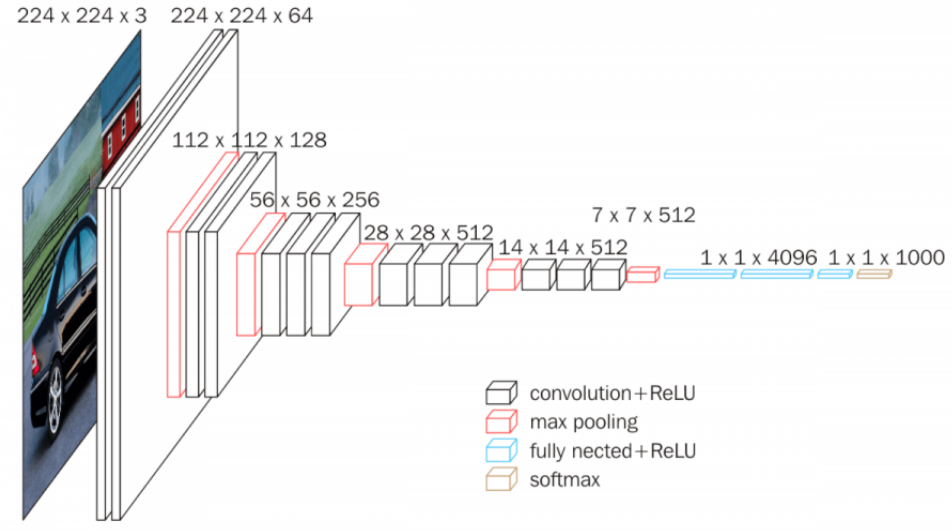

Figure 2.1: VGG-16

2.2 Model of Husain, Dellen, and Torras, 2016

Input to network is $224 * 224$ RGB image. This image is extracted as a frame from video for which we want to identify the action (Type of tennis stroke). We concatenated last max-pooling layer [1] of n [2] VGG16 networks and removed following layers including softmax.

[1] last red colored layer of $7 \times 7 \times 512$ dimension shown in figure 2.1
[2] number of frames extracted from single video

After removing layers we introduced a convolution $3D$ layer ($256 \times 3 \times 3 \times 3$), two fully connected layers of size 4096 and 1024 each followed by dropout layers. At end softmax layer of size K [3] is connected. Schematic representation is described in Figure 2.2.

2.2.1 Our implementation

For simplicity of understanding as well as implementation we divide network in three parts: A, B and C respectively.

- **Part A** comprises of n VGG16 models containing layers up-to last maxpooling layers.
- **Part B** comprises of concatenated feature maps of all VGGs of part A. This feature map has dimension of $512 \times n \times 7 \times 7$.
- **Part C** comprises of Conv$3D$ layer followed by fully connected, dropout and softmax layer respectively.

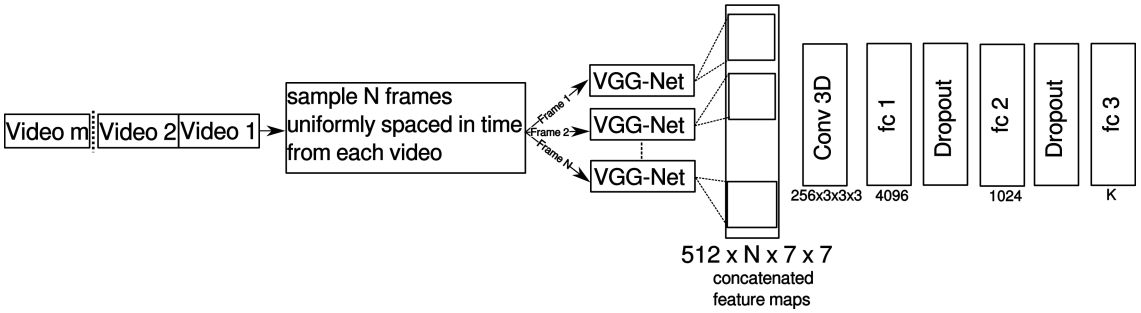

Figure 2.2: Illustration of the network

[3] number of classes to identify

3 Dataset

We are using THree dimEnsional TennIs Shots (THETIS) dataset by Gourgari et al., 2013 for training of our network.
THETIS is a sport based human action dataset comprised of the 12 basic tennis shots captured by Kinect. Tennis shots performed by 31 amateurs and 24 experienced players. Each shot has been performed several times resulting in 8734 (single period cropped) videos, converted to AVI format. The total duration of the videos is 7 hours and 15 minutes. The data are provided in 5 different synchronized forms (RGB, silhouettes, depth, $2D$ skeleton and $3D$ skeleton) The shots performed are given bellow:

- Backhand with two hands
- Backhand
- Backhand slice
- Backhand volley
- Forehand flat
- Forehand open stands
- Forehand slice
- Forehand volley
- Service flat
- Service kick
- Service slice
- Smash

As far as background is concerns, it is not static since most of the times it varies from one shot to the other, containing people acting in different ways in background. Example of background variations are illustrated in Figure 3.2. It contains very similar to each other actions, offering the opportunity to algorithms dedicated to gaming and athletics, to be developed and tested.

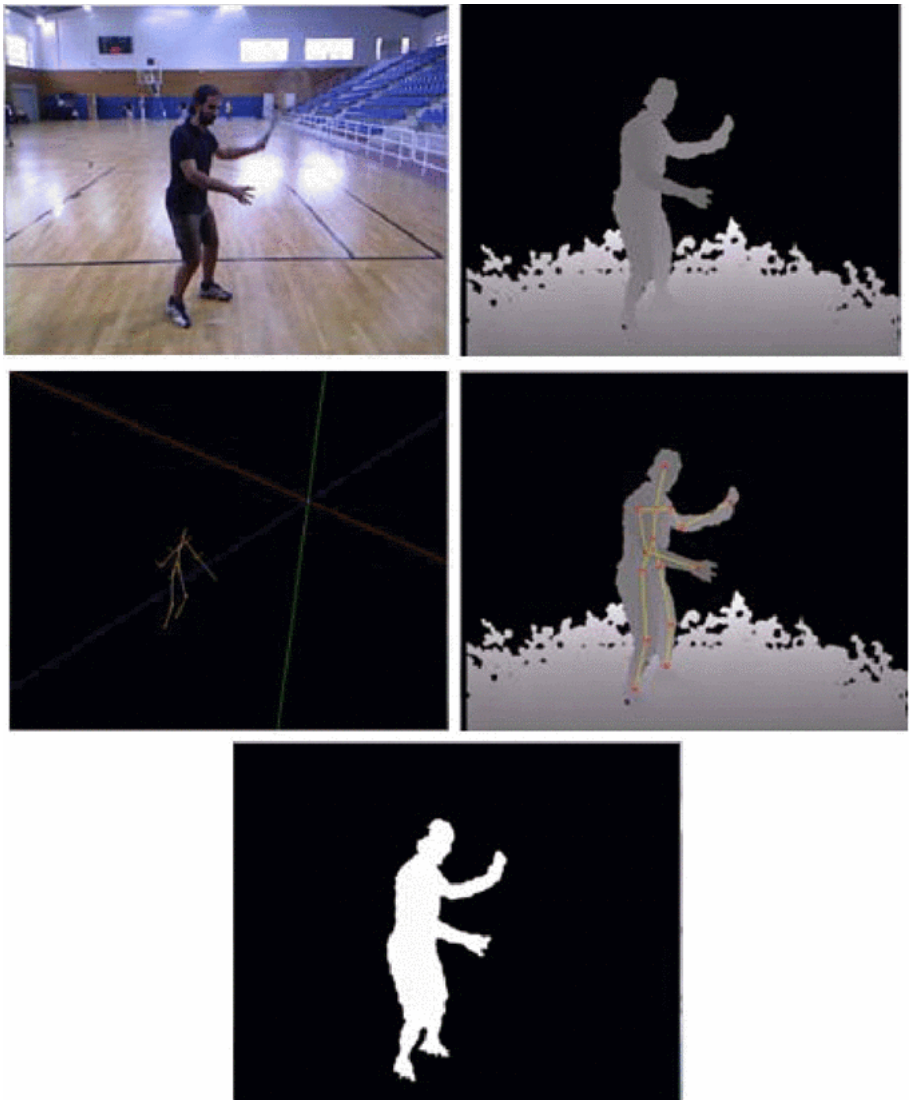

Figure 3.1: Different video formats of THETIS

Figure 3.2: Background variation in THETIS

4 Training and Testing

4.1 Hyperparameters

- **Algorithm:** Stochastic gradient descent
- **Loss function:** Cross entropy
- **Learning rate:** 0.001
- **Epochs:** 20
- **Initialization of network weights:** Pre-trained network
- **Number of training videos:** 150 (each category)
- **Number of testing videos:** 15 (each category)

4.2 Pre-processing

First we extracted frames from videos (RGB videos), after that re-sizing process done on these frames to make their dimensions 224×224. We choose 4 frames per video to feed into network, selection of frame done at equidistant e.g. if there are 20 frames then we pick $1, 4, 8, 12$ and 16 number frames respectively.

4.3 Selection of videos

Unlike training network with videos of same category until we exhaust all videos under that category and then move onto second category, we choose different videos from different categories in random fashion.

5 Comparison and Related Work

5.1 Comparison

Algorithm	Dataset	Accuracy
LRCNN (Dibua, n.d.)	THETIS RGB	82.3
STIP (Gourgari et al., 2013)	THETIS Depth	60.23
Model of Husain, Dellen, and Torras, 2016	**THETIS RGB**	**56.11**
STIP (Gourgari et al., 2013)	THETIS Skelet3*D*	54.4
Optical flow + LSTM (Dibua, n.d.)	THETIS RGB	53.2
Deep LSTM (Mora and Knottenbelt, 2017)	THETIS RGB	47.22

5.2 Related Work

For similar work as ours, we consider mainly three works done on THETIS so far:

- Dibua, n.d.
- Gourgari et al., 2013
- Mora and Knottenbelt, 2017

Dibua, n.d. uses two techniques. One method employs pre-trained CNN and second method captures optical flow. They feed these features into RNN networks with LSTM units in order to compare how well each approach classifies videos of players performing tennis strokes. Their work gives best results by feeding features generated from a pre-trained CNN into a many-to-many LSTM network, and averaging the softmax output to classify

videos.

Gourgari et al., 2013 uses the same procedure as in Laptev et al., 2008 except they use an extension of Harris 3D detector proposed by Laptev, 2005 which detects spacio-temporal points of interest and calculates the spacio-temporal descriptors, Histograms of Oriented Gradient (HOG) and Histograms of Optical Flow (HOF).

Mora and Knottenbelt, 2017 combines Inception model by Szegedy et al., 2015 and LSTM by Hochreiter and Schmidhuber, 1997. In their model videos are represented as sequences of features, extracted using the well-known Inception neural network, trained on an independent dataset. Then a 3-layered LSTM network is trained for the classification.

6 Limitations

- According to Mora and Knottenbelt, 2017 videos in the THETIS dataset do not contain the tennis ball, and this could explain why smash and serve are often confused. Another source of confusion are slices and volleys, both in backhand and fore-hand; these two actions are also quite similar for a human observer. Again, one main difference between volley and slice is that in the former the ball is hit before bouncing.
- Creators of THETIS dataset are not providing labels in their dataset which helps us in distinguishing between armature and player. If it is provided then our network has enough potential to identify between expert player and amateur one.
- Because of unavailability of resources (mainly GPU), we were able to deploy only four VGGs in parallel. Due to this limitation we could not provide more than four frames per video to network. This compromises performance of network.

7 Conclusion and future work

Our experiment as well as work of Husain, Dellen, and Torras, 2016 shows that this network performs with better accuracy on UCF101 and THETIS compare to LSTM. This provides us enough motivation to do explicit comparison of these model with LSTM and its variants for task of action recognition.

So far we observed others approach to use LSTM with action recognition involves

- Feeding data directly to LSTM/its variants.
- To perform feature extraction by other neural network architectures and then feed it to LSTM/its variants.
- To perform data processing (like optical flow) and then feed it to LSTM/its variants.

Our future work will be based on applying similar approaches to this network as listed above for LSTM/its variants.
Also problems such as undisclosed network hyper-parameters, network code and resource limitations it is hard to reproduce same network. We will try to reproduce researchers work, after that we are looking ahead to modify and create network's variants.

8 Appendix

Code for extracting frames from videos:

```
import os
import shutil
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pickle

def extractFrames(pathIn, pathOut):
    """
    This code takes absolute path of the video(pathIn) and returns the frames of the video in the folder pathOut.
    If the folder is not present, it will be created.
    """
    os.makedirs(pathOut, exist_ok=True)

    cap = cv2.VideoCapture(pathIn)
    count = 0

    cap.read()
    while (cap.isOpened()):

        # Capture frame-by-frame
        ret, frame = cap.read()

        if ret == True:
            print('Read %d frame: ' % count, ret)
            cv2.imwrite(os.path.join(pathOut, "{:d}.jpg".format(count)), frame)  # save frame as JPEG file
            count += 1
        else:
            break

    # When everything done, release the capture
```

```
    cap.release()
    cv2.destroyAllWindows()


# In[6]:


def extract_dataset(folder_name = '/user1/student/mtc/mtc2017/cs1725/youdata_training/',frame_dir = '/user1/student/mtc/mtc2017/cs1725/Extracted_Frames_ucf_youtube_train/',N=4):
    """
    folder_name contains the path to training folder.
    frame_dir contains the folder where the extracted frames of the videos will be stored.
    N is the number of frames we need from each video.
    """
    list_ = []
    list_ = os.listdir(folder_name) #contains name of all the labels
    #print('list_',list_)
    dict_of_labels = {} #stores the path to the extracted frames of an video as key and the label as value.
    #list stores class names
    for i in list_:
        tmp = folder_name + '/' + i # i is the label of video
        #print('i = ',i)
        _list = os.listdir(tmp) # stores the name of the videos in the class.
        for vid in _list:
            pathIn = tmp + '/' + vid
            #print('tmp - vid ',tmp,vid)
            pathOut = frame_dir + i + '_' + vid + '_jpg'
            dict_of_labels[pathOut] = i
            print('pathin-out',pathIn,pathOut)
            # Extracting frames from the video and storing to the required destination

            extractFrames(pathIn,pathOut)
            # To select the frames we need
            list_of_files = os.listdir(pathOut)
            num_frames = len(list_of_files) # counts the number of frames
            selected_frame_indices = np.linspace(start=0,stop=num_frames,num=N+1,dtype=np.int)[:-1]
```

```
            selected_frame_names = [str(x) + '.jpg' for x in
    selected_frame_indices]
            print(selected_frame_names)
            # Deleting the unnecessary frames
            for file in list_of_files:
                if file in selected_frame_names:
                    print('the following file remains', file)
                else:
                    print('this should be deleted:', file)
                    os.remove(os.path.join(pathOut,file))

        #print(_list)
    return dict_of_labels


# In[8]:


def dict_save(framelist, path = '/user1/student/mtc/mtc2017/cs1725
    /',file = 'dict.save'):
    """
    Utility function To save the dict_of_labels in a file for
    future use.
    """
    with open(path+file, 'wb') as f:
        pickle.dump(framelist, f)

def dict_load(path = '/user1/student/mtc/mtc2017/cs1725/', file =
    'dict.save'):
    """
    Utility function To load the dict_of_labels from a file for
    future use.
    """
    with open(path+file, 'rb') as f:
        framelist = pickle.load(f)
    return framelist


# In[9]:


train_folder_name = os.path.join(os.getcwd(),'Train/')
train_frame_dir = os.path.join(os.getcwd(),'Extracted_Frames_train
    /')
```

```
105 print(train_folder_name,train_frame_dir)
106 test_folder_name = os.path.join(os.getcwd(),'Test/')
107 test_frame_dir = os.path.join(os.getcwd(),'Extracted_Frames_test/'
    )
108 print(test_folder_name,test_frame_dir)
109
110
111 # In[11]:
112 dict_train = extract_dataset(train_folder_name,train_frame_dir)
113 dict_save(dict_train,os.getcwd() + '/',file='dict_train.save')
```

Bibliography

Dibua, Vincent Chow Ohi. "Action Recognition in Tennis Using Deep Neural Networks." In: (cit. on p. 12).

Gourgari, Sofia et al. (2013). "Thetis: Three dimensional tennis shots a human action dataset." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 676–681 (cit. on pp. 8, 12, 13).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory." In: *Neural Comput.* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 13).

Husain, Farzad, Babette Dellen, and Carme Torras (2016). "Action recognition based on efficient deep feature learning in the spatio-temporal domain." In: *IEEE Robotics and Automation Letters* 1.2, pp. 984–991 (cit. on pp. vi, 12, 15).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*, pp. 1097–1105 (cit. on p. 5).

Laptev, Ivan (2005). "On space-time interest points." In: *International journal of computer vision* 64.2-3, pp. 107–123 (cit. on p. 13).

Laptev, Ivan et al. (2008). "Learning realistic human actions from movies." In: *CVPR 2008-IEEE Conference on Computer Vision & Pattern Recognition*. IEEE Computer Society, pp. 1–8 (cit. on p. 13).

Mora, Silvia Vinyes and William J Knottenbelt (2017). "Deep learning for domain-specific action recognition in tennis." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, pp. 170–178 (cit. on pp. 12–14).

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (cit. on p. 5).

Szegedy, Christian et al. (2015). "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9 (cit. on p. 13).

Wang, Jindong et al. (2019). "Deep learning for sensor-based activity recognition: A survey." In: *Pattern Recognition Letters* 119, pp. 3–11 (cit. on p. 4).