Recognising Eating Actions

Minsung Kim

A Dissertation
Submitted to the School of Informatics
Supervised by Prof Robert B Fisher

School of Informatics
University of Edinburgh
April 2022

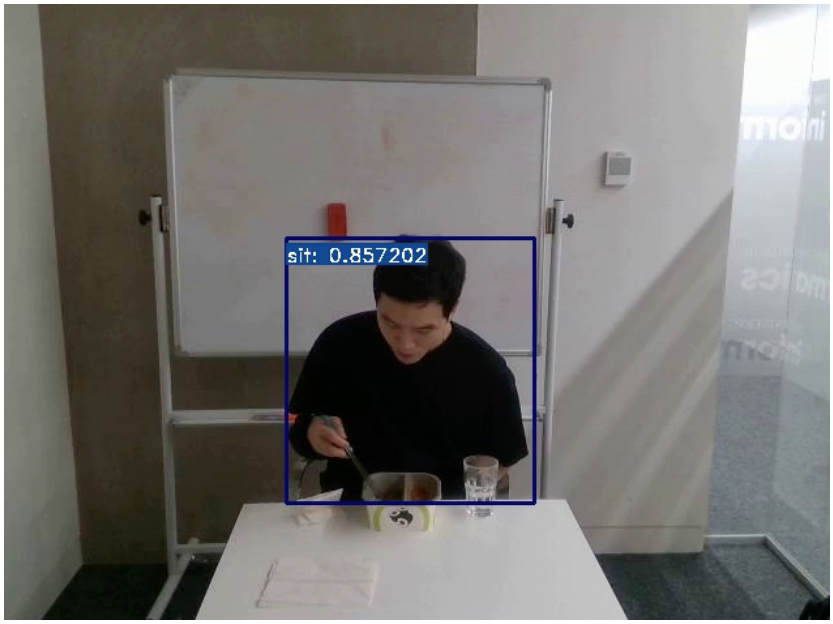

Figure 1: Spatio-Temporal Action Recognition[1]

Figure 2: Motion without wrist weights

Figure 3: Motion with wrist weights

Abstract

Artificial Intelligence(A.I.) has advanced tremendously and it has become easy to find A.I in everyday life, which improves human well-being. This includes the healthcare field.

In this dissertation, we investigate the potential to improve elders' well-being by implementing a surveillance camera system based on computer vision. This camera will monitor the motor movement of elders to judge whether they are still able to live alone, by using action-based and joint-motion analysis-based algorithms, such as Temporal Pyramid Network (TPN), as applied to eating behaviour. We aimed to make the algorithms classify eating motions from videos or images, so we can use the algorithms for the next step, which is to identify motor deterioration in the elders.

For training and testing, we used custom data sets. We recorded our eating motions with and without wrist weights to imitate an elderly person's physical strength. Datasets were divided into groups and the groups were mixed to determine how effectively and accurately models generalize and classify actions.

The test results showed that TPN works well when enough training data is present, but it seemed TPN does require a minimum amount of data relating to the target, whose actions are being classified. The test results were also compared with the test results when using TIN and OmniSource.

Finally, we discussed the project and the plan for the next project. We also talked about the problems, which occurred in this project, and possible future solutions.

Contents

Chapter 1

Introduction

1.1 Motivation

There have been huge achievements in the medical field, which have lead to a significantly extended human life expectancy. According to the World Economic Forum, the estimated life expectancy at birth has increased by 20 years in developed countries and 30 years in developing countries, in the last 70 years. These figures have been predicted to continue to rise, as shown in Figure 1.1.

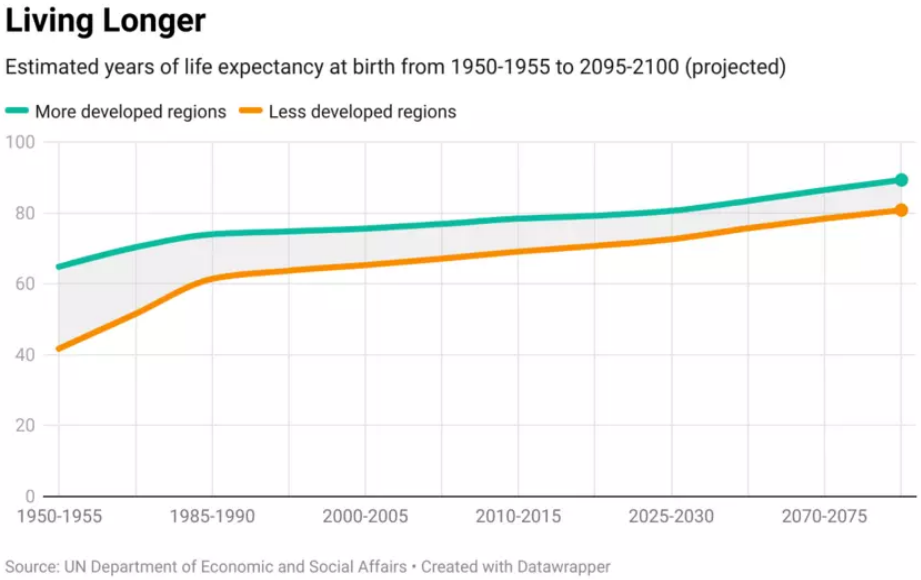

Figure 1.1: Life expectancy[2]

On the other hand, the number of young people has decreased in many countries. For example, the population of South Korea has been decreasing since 2020, because the birth rate of Korea has been less than 1 for several years now (see Figure 1.2).

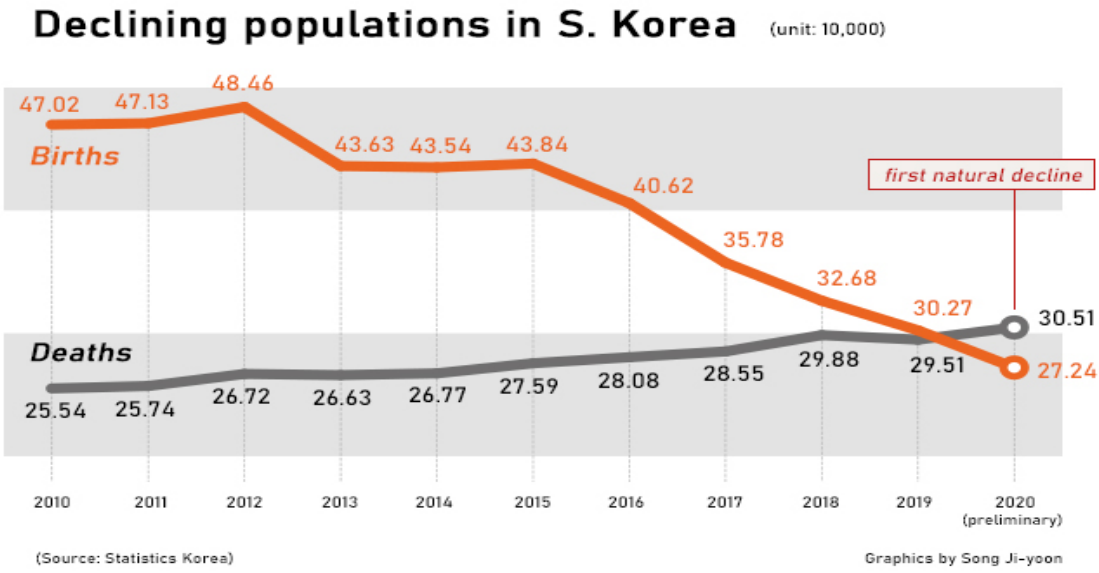

Figure 1.2: South Korea population declining(unit: 10,000 people)[3]

These social phenomena cause new problems, including a decreasing medical workforce. There will be fewer doctors and nurses, and there will be more people in old age who need medical care.

Many elders suffer from pathological diseases. According to Medical News Today, menopause causes a decrease in estrogen, and this can lead to osteoporosis. Due to this condition, a person's body makes too little new bone, loses too much bone, or both[4]. Elderly people also fall easily because of vision loss, balance problems, dementia, heart disease, etc. Combined with osteoporosis, there is a significant risk that a fall could lead to serious injuries.

The diseases listed above illustrate examples of cases in which elders require constant care, which means it is then often difficult for them to live in their homes, since it is not easy to find someone to care for them full time. It is also often financially unrealistic for most people. Moreover, patients do not want to relocate to care homes, because of the stress associated with entering a new, strange environment[5]. Many of the problems addressed above can be prevented if they can continue to live in their own homes for longer, with some additional monitoring, and also if timely action is taken by observing any drastic changes in markers such as pulse and posture[6].

1.2 Problem Definition

Traditional products to check health status give accurate results, since they are attached to the human body, but elderly people often forget to wear, or do not want to wear, these products. A camera-based system avoids this problem.

Eating is inevitable for almost every organism to survive - humans are no exception. Human eating habits are quite hard to change because this action is a primal

instinct for self-preservation and closely connected to a human's brain. People who are on a diet often face difficulties when they are stressed or tired because their cortex is losing control and their instincts are kicking in[7]. Therefore, observing eating seems a good strategy to detect abnormalities in an elder's health, since eating is a universal behavior, and it is easy to check for abnormal actions, as eating habits do not change often.

This system could detect changes in a person's eating motions to check whether the person requires medical support. Because eating is a universal motion, this system can be applied to everyone. The system investigated in this dissertation will track the movement of joints and identify the activity. [6].

1.3 Research Questions

The research focused on the following 3 research questions in the context of the TPN approach.

- How accurately does the system classify actions?
- How well can the system generalise its classification? Does the system trained with a certain group maintain accuracy when it is tested on other groups?
- Can the system recognise a new person well by adding a small amount of training data acquired from them? Does this cause the system to display higher accuracy than the aforementioned scenario?

1.4 Summary

To create the surveillance system and answer the research questions, we needed to figure out how to track and classify body movement using computer vision. We explored several methods, such as motion analysis, for instance. Motion analysis consists of pose estimation, pose tracking, and motion estimation. More details are in Chapter 2.

Since there is no specific dataset for the motion of a human eating, we needed to prepare it ourselves. We recorded our eating motion with or without wrist weights to simulate elders' physical strength, as shown in Figure 2 and Figure 3. After this, we labeled the data with eating action classes by using an open source annotation tool. We discuss this in Chapter 3.

To experiment with the obtained dataset, we set up the environment first. We used an open-source toolbox to train and test models. We also changed the configuration of the toolbox for data pre-processing, hyperparameters and cross validation. We tried to run the toolbox on different machines to find the machine with the shortest training time. See Chapter 4.

We gathered test results including Top1 and Top2 accuracy, and predictions for each experiment. We also compared our results to the TIN and Omnisource algorithms,

and concluded ***. Related charts and graphs can be found in Chapter 5.

Using all of the information above, we discuss our conclusion in Chapter 6. This chapter contains the scope of our work, what was and was not effective, and opportunities for future research.

Chapter 2

Literature Review

One approach to vision-based motion capture uses markers to track human body movement. Despite it being well-established as a method which yields the most accurate results, it requires a controlled situation, so it would not find universal application[8]. On the other hand, marker-less systems are considered to be free from these constraints[6][8]. It is, however, quite hard to segment motions from videos, since this requires prior information, such as the background image [6][9].

There exists the possibility of losing critical information while extracting the shape of the human body from the video, if the person is not facing the camera. Accuracy could be improved by using dedicated algorithms and models for specific poses, but this is not common due to cost inefficiency.

There have been improvements, which have made it possible to overcome these problems, and we will discuss these while explaining camera-based motion analysis with literature reviews in three steps: Pose Estimation, Pose Tracking, and Activity Recognition.

2.1 Pose Estimation

Pose estimation is the process by which the position of limbs or body parts is extracted from videos or images[6]. Before deep learning became popular, the Pictorial Structure Framework (PSF) was used for Human Pose Estimation (HPE). PSF is based on a statistical framework, but it has some limitations - it could not capture and model hidden or poorly visible limbs[10].

In the past, deep learning was not developed enough to automate this pose estimation process, so manual work was necessary. However, because of the advanced pose estimation algorithms, now it can be done by PyTorch-based tools without much effort. There are two types of pose estimation: 2-Dimensional (2D) and 3-Dimensional (3D).

These categories can be separated further by different body modelling techniques such as kinematic, planar, and volumetric, or target number such as single person or multi-person[6][11].

2.1.1 2D Pose Estimation

2D Pose estimation algorithms extract patterns and representations from a video or an image, and estimate the pose by using the features. These features contain spatial joint locations. At first, Toshev et al. proposed the Convolutional Neural Network (CNN) based approach called DeepPose[12]. Even though DeepPose showed good results, it struggled to estimate multi-human outcomes. If there are many people, the complexity increases. This causes the inference time in real-time to increase. To solve these problems, Top-down and Bottom-up methods were developed[10]. Top-down locates people, followed by estimating body parts and pose. This method was used by Li et al. at AlphaPose[13]. Bottom-up locates body parts first, then calculates the pose. Zhe et al. used this approach in OpenPose[14].

2.1.2 3D Pose Estimation

3D pose estimation is similar to 2D pose estimation, but targets are found in a 3D space. To estimate poses in a 3D space, we need more information than the RGB color model data offers, which is depth data. There are two types of range data representations, which are range image and point cloud. Even though depth data gives us more accurate pose estimation, we did not investigate further since we decided to use 2D pose estimation methods.

2.2 Pose Tracking

Figure 2.1: Pose tracking from our dataset

There are many occasions in which pose estimation is not available - such as an image or a video is occluded or the target's body parts are blocked by some obstacles. Pose tracking makes it possible to track the target's body through assumptions about its next

locations, by using deep learning techniques. In real life applications, it is not always possible to get clear images or videos of the target, so pose tracking proves to be quite an interesting subject. There are two different types of tracking algorithms - Correlation Filter Trackers (CFTs) and Non-Correlation Filter Trackers (NCFTs)[6][15].

CFTs use the correlation filter to separate a target and background from an image and a video. On the other hand, NCFTs use different methods such as Multiple Instance Learning(MIL), matching mechanism, etc.[6].

2.3 Activity Recognition

Activity Recognition identifies the actions and intentions of one or more persons from videos or images of them. It could be broadly used in many fields, i.e. visual systems for active and assisted living, smart home systems, and healthcare monitoring systems[16]. There are two types of activity recognition sources: trimmed videos and untrimmed videos. While untrimmed videos include more than one kind of actions, or none, trimmed videos only include one action - i.e., untrimmed videos consist of walking and talking actions, and trimmed videos only have talking actions. Since recognising eating actions is the main goal of this paper, we are going to discuss several behaviour classification algorithms designed for trimmed videos, such as TPN, TIN, and OmniSource[17][18][19].

2.3.1 Feature Extraction

Similar to the steps we mentioned above, Activity Recognition also uses a feature extraction method. The difference here is that Activity Recognition uses spatial or temporal signals to extract features. In 2D based algorithms, spatial features can be obtained through body representations from the image. For temporal features, we parse an action into pieces and make groups depending on how the pieces are similar to each other[20]. In the past, either spatial or temporal features were used for algorithms, but nowadays both features are combined as spatio-temporal features to be used in algorithms, such as TPN.

2.3.2 Feature Based Learning

After extracting features, we need to learn from them. There are several methods to do this, such as genetic algorithms and deep learning. A genetic algorithm is about creating functions from random features from the feature pool and crossing them over, and eliminating the combinations with less accuracy until reaching certain epochs.

Deep learning has become the most popular method for feature based learning, which can be divided into supervised and unsupervised models. For supervised models, the system is trained with example inputs and desired outputs in order to learn a general function that maps inputs to outputs. For unsupervised models, the system is only presented with unlabeled data and the goal is to infer a certain structure.

2.3.3 Algorithms

Many action recognition algorithms have been developed recently with the help of deep learning. One of the most interesting aspects of deep learning is that deep learning models consist of layers. This characteristic allows the models to have not only a variety of structures, but also the flexibility to be mixed with new elements easily. The algorithms such as TPN, TIN, and OmniSource have similar characteristics. They use existing layers, such as ResNet, as a backbone, and manipulate layers to make their own algorithms.

Since the standards for algorithms are unclear, researchers tested algorithms with different datasets and setups. Therefore, the comparison between algorithms might not be accurate, due to the presence of different variables.

TPN

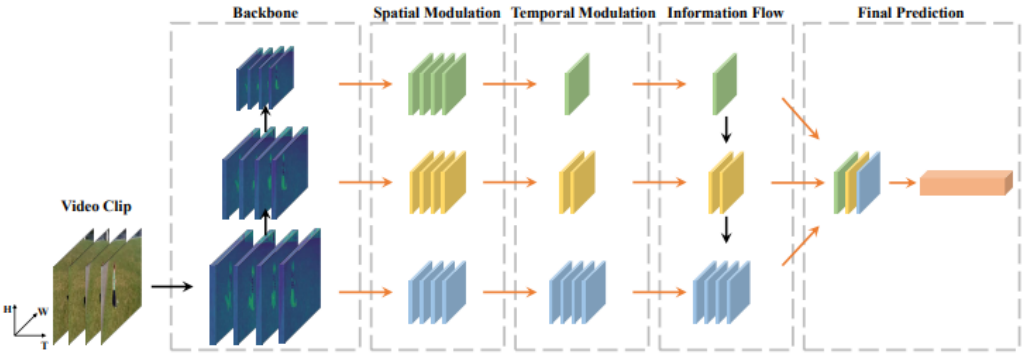

Figure 2.2: Framework of TPN (Figure copied from [17])

Visual tempo describes how fast an action occurs. Actions can be differentiated by visual tempo. For example, walking, jogging, and sprinting share high similarities in terms of visual appearance, but their speeds are different. TPN models visual tempos of different actions to recognise and differentiate between them. TPN works as an auxiliary module, so it could be combined with many existing action recognition models.

TPN uses two essential components: the source of features and the fusion of features. These two form a feature hierarchy for the backbone[17]. TPN also makes a frame pyramid. Each pyramid level samples the input frames at a different temporal rate, i.e., a 2-level pyramid for 64 frames has 16 and 2 intervals, so each level samples 4 frames and 32 frames. At each level, different backbone sub-networks use the frames, and the outputs are gathered together to make the final predictions. Sampling different frames allows TPN to represent the actions at different visual tempos.

TPN seems to work well in general, but it shows most of its improvements when visual tempos of the action classes are much different. With the kinetics-400 dataset, TPN-ResNet101 had 78.9% Top-1 accuracy and 93.9% Top-5 accuracy, while other

algorithms with same or deeper backbone depth had from 68.7% to 77.8% Top-1 accuracy and 88.5% to 93.3% Top-5 accuracy. Furthermore, TPN with ResNet50 had 77.7% Top-1 accuracy, so it seems TPN works well with the relatively shallow backbone networks[17].

TIN

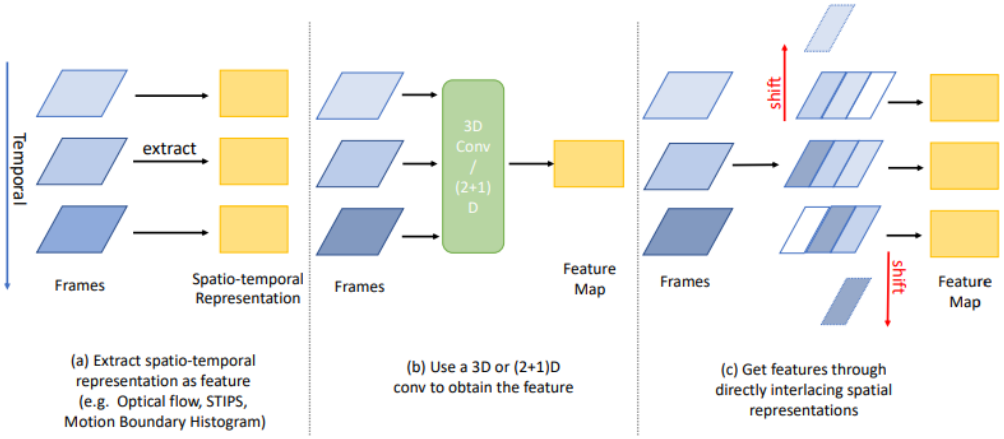

Figure 2.3: TIN pipeline(Figure copied from [18])

Unlike traditional approaches in the vision community, which compute spatio-temporal features separately, Temporal Interlacing Network(TIN) combines spatial and temporal information by mixing spatial features with those from the past to the future, and vice versa[18]. This is advantageous for computing resources, since TIN decreases 2-dimensional information to 1-dimensional information. There are three steps in this process. First, inputs must be split into groups and the offsets and weights from neighboring frames must be obtained. Second, the offsets should be applied to their groups by shifting and inserting the shifted feature into a temporal dimension. The third step is to concatenate the split features and make a temporal-based group by using the learned weights.

In the test results of the Something-Something v1 dataset, TIN showed 49.6% Top-1 and 78.3% Top-5 accuracy, while TPN showed 49.0% Top-1 and 62.0% Top-5 accuracy. TIN appears to have much higher Top-5 accuracy compared to TPN[17][18].

OmniSource

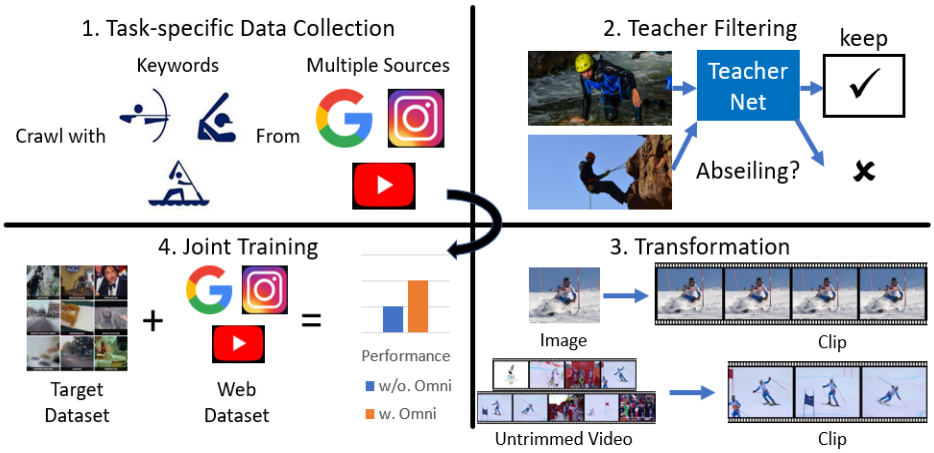

Figure 2.4: Pipeline of OmniSource (Figure copied from [19])

Unlike other algorithms, OmniSource leverages web data to train models. It is also not strict regarding the format of the web data, so the web data can be a video, image, etc. For the process, task-specific data collections curate data samples with multiple formats. Then, a teacher model filters data samples to make a unified form. Finally, a joint-training strategy deals with the domain gaps between multiple data sources and formats in webly-supervised learning.

With the Kinetics 400 dataset, OmniSource had 83.6% Top-1 and 96.0% Top-5 accuracy while TPN had 78.9% Top-1 and 93.9% Top-5 accuracy. Because OmniSource uses the irCSN-152 backbone and TPN uses the ResNet101 backbone, it would be interesting to see TPN using the same backbone as OmniSource[17][19].

2.4 Summary

Similar to other human actions, eating looks so natural to us because we do it everyday without giving it much thought. However, eating consists of many auxiliary actions, so it is not an easy task to fit it into a certain framework. For example, to eat porridge, the auxiliary actions can include picking up some of the porridge from a bowl with the spoon, moving the hand towards mouth, and finally eating the porridge.

Our project goal is to recognise the auxiliary actions through three steps - pose estimation, pose tracking, and activity recognition, which we mentioned in the literature review. We are going to use this result for the next step - detecting the steps of eating actions.

We used the three action recognition algorithms described in the literature review - TPN, TIN, and OmniSource for this project. In this dissertation, we focused on TPN specifically.

Chapter 3

Data Acquisition

As we mentioned in the introduction, there is no video-based dataset dedicated to eating motions. Therefore, we needed to prepare our own dataset. We recorded 2 eating videos for each undergraduate student, where one clip has the students wearing weight wrists, and the other not wearing the weights, in order to imitate elders' physical strength. We gathered a total of 6 sequences from 3 students. In addition to the data we recorded, there are also data collected by Ahmed Raza, who is the Ph.D student that proposed this project[6]. These data contain 25 sequences of Professor Robert Fisher, Ahmed, and Ph.D student Arushi Goel, eating with or without using wrist weights. For the experiments, we made 2 groups - group A and group B. Professor Fisher, Ahmed, Arushi are in group A and the undergraduate students belong in group B. The sequences were recorded with different frame per second(fps) rates as well. Most sequences were recorded with 30fps, but some sequences from group A and all sequences from group B were recorded with 15fps. These fps data were used for preprocessing data, which is mentioned in Chapter 4. We used an RGB-Depth camera for recording, which can capture RGB and depth data while recording videos. In this paper, we used the RGB data only, but depth data could be used in the future for further research.

Since most people tend to sit to eat, we assumed that elders do this as well. We used upper body poses only, because significant lower body movements appear infrequently, or not and all, while people are eating - also, a camera usually cannot see the lower body. The upper body pose can include the head, shoulders, elbows, and wrists.

We need to gather constant data from elders in order to detect signs of changes in behaviour, so in this case we assumed that elders will eat at the same location - in their home, with a camera present to record their eating. To imitate this environment, we shared a mostly common environment setup; a person sat at a table and ate or drank, indoors. We recorded our videos while we were using different utensils. For example, one student used chopsticks and other students used a spoon or their hands.

To train our model, we needed to define what eating actions come first. It is a crucial task, since we need a standard against which to judge what a person is doing. This surveillance system for elders should be equipped to deal with many random people - since humans generally have quite different eating behaviours, associated with

their gender, ethnicity, etc. According to grubHub, men are more likely to eat foods, which are more often consumed by hand, such as buffalo wings, while women tend to order foods which are more often consumed with the use of flatware/cutlery, such as avocado rolls[7]. In addition, East Asian people are more likely to use chopsticks than other people. To quantify these various circumstances, we needed to define actions as a combination of composite actions and sub-actions. Composite actions consist of sub-actions. For example, drinking is a composite action of the following sub-actions - picking up a glass, drinking, and putting the glass back.

As mentioned above, there are different ways to eat depending on the person, and it would be an obstacle to our research if we did not categorize these actions properly. There would be two main problems - misclassification and a lack of data. As people have different eating habits, they also have different perspectives on classification, meaning that people might classify actions differently, despite having the same actions presented to them. For example, it is common to eat soup with rice in South Korea. Some people mix rice with soup and drink it from a bowl while they are holding the bowl, but they consider this to be eating. However, people who use a spoon to eat soup from a soup dish would consider this drinking. Therefore, misclassification happens. As previously mentioned, without rigorous categorizing, we might lack data as well. If we try to make too many sub-actions, there might not be enough data to train for the model. For example, if we separate actions by which utensils people use, such as chopsticks, knives and forks, the chances of the model misclassifying actions is more likely, if the data were collected in the Western world.

To avoid the problems listed above, we went through several steps to make a sub-action list. It consisted of 33 sub-actions, which was then sized down to 10 sub-actions. We needed to merge some sub-actions due to a lack of samples. Also, some sub-actions were hard to distinguish visually, and some sub-actions were irrelevant to eating, so we merged or removed the actions. For labelling our videos, we used an open-source program called VGG Image Annotator(VIA) [21].

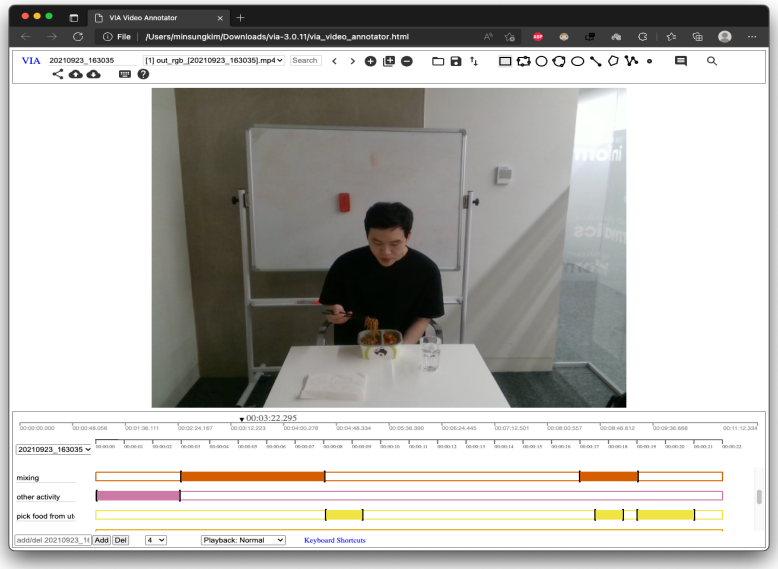

Figure 3.1: VIA in action

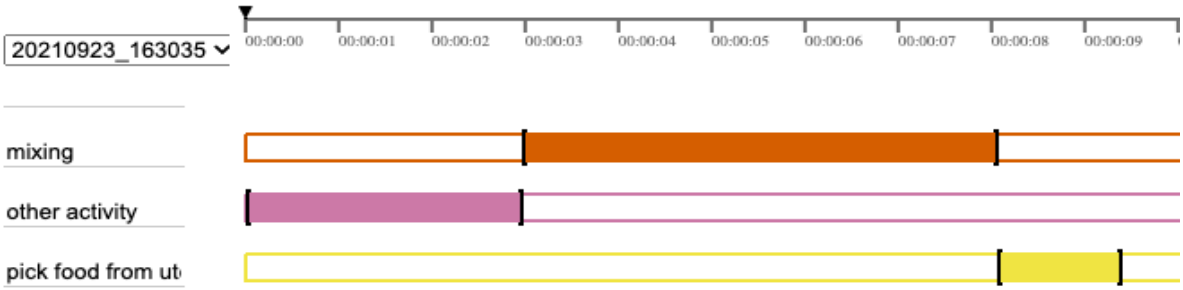

Figure 3.2: VIA timestamps

For each undergraduate student, we labelled two videos; one is with the weights and one is without the weights. Since there are no tools available to automate this process, we needed to label manually. As shown in Figure 3.1, first we entered the actions at the bottom left corner. After that, we played the video and set timestamps indicating that an action happened at that time period. For example, Figure 3.2 shows that 'other activity' happened between 0 to 3 seconds and 'mixing' happened between 3 to 8 seconds.

# CSV_HEADER	file_list	temporal_segment_start	temporal_segment_end	metadata
1_ObvouN1i	["out_rgb	0.002	0.981	{"20210923_163801":"other activity"}
1_1bHnqJGe	["out_rgb	0.981	2.377	{"20210923_163801":"pick up a tool w
1_9aHBdonY	["out_rgb	2.377	2.961	{"20210923_163801":"mixing"}
1_DGWt4D70	["out_rgb	2.961	5.856	{"20210923_163801":"pick food from
1_X8mUCxoc	["out_rgb	5.856	8.148	{"20210923_163801":"eat it"}
1_IB6ibwSP	["out_rgb	8.148	15.815	{"20210923_163801":"chewing"}
1_NpHy5d4o	["out_rgb	15.815	27.586	{"20210923_163801":"pick food from

Figure 3.3: Group B CSV label file

Overall, it took 3 hours to label videos 11 and 14 minutes in length. The labels created from this process were saved as Java Script Object Notation(JSON) and Comma-Separated Values(CSV) files. We used the CSV files for data preprocessing. The CSV files contain lists of headers, the start and end time of actions, and the name of actions. Table 3.1, 3.2, 3.3 show the statistics on the number of actions in the videos. The actions are tied to numbers; 'other': 0, 'pick up food from a container with both hands': 1, 'pick up food from a container with one hand': 2, 'move hand towards mouth': 3, 'eat food': 4, 'move hand away from mouth': 5, 'chewing': 6, 'no action': 7, 'pick up food from a container with tools in both hands': 8, 'pick up food from a container with a tool in one hand': 9. We can see that group B does not have certain actions. This happened because 'move hand towards mouth' and 'move hand away from mouth' were not in the action list when we labelled the videos. Also, since no undergraduate students used tools in both hands, 'pick up food from container with tools in both hands' appeared as 0.

Action No.	Action	Count
0	Other	266
1	pick up food from a container with both hands	152
2	pick up food from a container with one hand	208
3	Move hand towards mouth	393
4	Eat it	349
5	Move hand away from mouth	384
6	Chewing	240
7	No action	5
8	Pick up food from a container with tools in both hands	119
9	Pick up food from a container with a tool in one hand	33
Total		2,149

Table 3.1: Group A list of actions and its count

Action No.	Action	Count
0	Other	307
1	Pick up food from a container with both hands	19
2	Pick up food from a container with one hand	84
3	Move hand towards mouth	0
4	Eat it	172
5	Move hand away from mouth	0
6	Chewing	119
7	No action	48
8	Pick up food from a container with tools in both hands	0
9	Pick up food from a container with a tool in one hand	99
Total		848

Table 3.2: Group B list of actions and its count

Action No.	Action	Count
0	Other	573
1	Pick up food from a container with both hands	171
2	Pick up food from a container with one hand	292
3	Move hand towards mouth	393
4	Eat it	521
5	Move hand away from mouth	384
6	Chewing	359
7	No action	53
8	Pick up food from a container with tools in both hands	119
9	Pick up food from a container with a tool in one hand	132
Total		2,997

Table 3.3: Group A + B list of actions and its count

Chapter 4

Methodology

Once we gathered the data and labelled them, we needed to make the data trainable for our algorithm. This process is dependent on frameworks, because they use different ways to read labels and files. Here we used an open-source toolbox based on PyTorch, named MMAction2. We did 5-fold cross-validation to ensure the result was reliable. TPN is already included in MMAction2, so we did not need to implement the algorithm. After data preprocessing, the algorithm trained the model by collecting hierarchical features and aggregating them to calculate weights. The training was performed 5 times for each configuration. In the testing, we used the saved weights from the training to predict classes, and we gathered several kinds of statistics including top 1, top 5, and mean-class accuracy for evaluations. Further details are described in each respective section.

4.1 Data Preprocessing

4.1.1 Data Cleaning

# CSV_HEADER	file_list	temporal_segment_start	temporal_segment_end	metadata
1_nyxNyYiM	["out_rgb_[20210923_155848].mp4"]	47.084		{"20210923_155848":"other activity"}
1_6S9uj0A6	["out_rgb_[20210923_155848].mp4"]	0.018	3.502	{"20210923_155848":"other activity"}
1_doGUa0fd	["out_rgb_[20210923_155848].mp4"]	3.502	8.919	{"20210923_155848":"pick food from
1_oJY33P4e	["out_rgb_[20210923_155848].mp4"]	8.919	11.086	{"20210923_155848":"eat it"}

Figure 4.1: Wrong label

After labelling the videos, we needed to check whether the labels were genuine, for the training. First, we needed to check the labels are in the correct format. For example, in Figure 4.1, there is an empty cell in the second row, which is for the end time of the action. Since the start time of actions has to be smaller than the end time of actions (because actions cannot be done before they start), this entire row had to be removed. We removed it by comparing the start time and the end time, and checking the type of inputs, so if it is none or not integer, the row was removed.

There were also problems with the videos themselves. The videos were saved as image and depth files, and we used the image files for our project. However, some videos were corrupted, so a chunk of images were missing from the videos. Therefore, we needed to remove the row which does not have enough corresponding images from the start time to the end time. This process was done when we created the Rawframe dataset.

4.1.2 Rawframe Dataset

```
1  /content/datasets/2_h39cab8N 15 5
2  /content/datasets/2_DfTBiGcP 16 5
3  /content/datasets/2_MNNCwfl0 17 5
4  /content/datasets/2_dOXf7qYc 6 5
5  /content/datasets/2_PekYzMIs 27 5
6  /content/datasets/2_qL3Sw2Zp 13 5
7  /content/datasets/2_Qe51rQTG 8 5
8  /content/datasets/2_k4fIbdLM 12 5
9  /content/datasets/2_WBlHEatE 16 5
10 /content/datasets/2_kDbBDpB5 17 5
```

Figure 4.2: Rawframe annotation

As was mentioned, frameworks support different formats of training, testing, dataset, and annotation style. MMAction2 uses 3 types of dataset; Rawframe, Video, and ActivityNet. Here TPN can use only Rawframe dataset, which consists of folders, and each folder contains images of an action. Each dataset also has its own annotation style, and the format of a Rawframe dataset annotation is shown in Figure 4.2. Every line has 3 parts of a string - data location, total frame, and action number. Data location indicates where the image folder is. Total frame indicates how many images are stored in the folder. Action number shows which one of the 10 actions we defined is in the folder. We needed to follow this format, so we extracted the group B CSV files to get CSV headers, start time, end time, and actions from the CSV files, and created label text files using python scripts.

Project	Action	Imgs	head_x	head_y	head_z	Mid-Shoulder_x	Mid-Shoulder_y
20210804_145925	5	625	0.0567485913634300	0.3936030268669130	2.2880001068115200	0.0129989571869373	0.3215812742710110
20210804_145925	5	626	0.0566245764493942	0.3927428722381590	2.2880001068115200	0.0130312517285347	0.3223802149295810
20210804_145925	5	627	0.0607155710458756	0.3953233063220980	2.2880001068115200	0.0170262716710567	0.3238449990749360
20210804_145925	5	628	0.0607155710458756	0.3916015923023220	2.293000102043150	0.0170262716710567	0.3199062049388890

Figure 4.3: Group A CSV file

CSV files for group A have a different CSV format to the one shown in Figure 4.3; different column names, each row representing each frame, containing more data, e.g. joint positions. So, we needed to use another script to extract the information.

The next step was creating the image folders and storing frames in the folders. Here, we also needed to use different scripts due to different annotation formats. For group A, each sequence was provided as a compressed file with the name corresponding to the 'Project' column in Figure 4.3. We used 'Action' 'Imgs' to get image files from the compressed file and saved them in each folder, named as its 'Project' column. For group B, the process was almost the same to that for group A, but we needed to convert the start time and end time to frames, since the values were saved in seconds. The frame list for the videos was provided by Ahmed as a dictionary, {'20210509_233500':15, ..., '20210603_130948':30} where keys are the sequences and values are frames.

4.1.3 Cross-Validation

```
# 5-fold cross-validation
# Rawframe contains 5 lists of Rawframe data
def 5fold(Rawframe):
    # iterate 5 times
    for i in range(5):
        # fold = [0, 1, 2, 3, 4]
        fold = [*range(5)]
        train = []
        # add label data at index i to val
        val = label[i]
        # mod 5 in case i=4 to prevent an out of range problem
        test = label[(i+1)%5]
        # remove val and test from train
        fold.pop(i)
        #
        fold.pop(i%4)
        # fold = [2, 3, 4] when i=0
        for i in fold:
            # add label data at index i to train
            train.append(label[i])

        write_annotation(train, val, test)
```

Listing 4.1: 5-fold cross-validation

For the sake of data reliability and to prevent overfitting, we used a 5-fold cross-validation method. For group A, Ahmed provided 5 CSV files representing the whole of the data split into 5 pieces, so once we extracted the Rawframe data from the CSV files, we only needed to run the code 4.1 for 5-fold cross-validation. This code iterated 5 times to create 5 sets of train, val, and test annotation files, by using the extracted Rawframe data. For example, there were 5 train annotation files which were separated by split numbers, e.g. 'train_sp1.txt' ... 'train_sp5.txt'. val and test followed. Table 4.1 shows the result after running the code.

For group B, there was no CSV file for cross-validation, so we needed to separate our data into 5 sets first. To make sure the data are evenly distributed, we checked two things when processing a sequence; who the person is and the number of actions taken in the sequence. Even though we have 2 kinds of sequences for each undergraduate student; with or without weight wrists, we consider them as the same kind of data because we do not have enough data to split into small groups. Therefore, for each

Rawframe data A, B, C, D, E			
Split	Train	Val	Test
1	C, D, E	A	B
2	A, D, E	B	C
3	A, B, E	C	D
4	A, B, C	D	E
5	B, C, D	E	A

Table 4.1: Cross-Validation sets example

person, we counted how many numbers of actions happened in their two videos, and split them into 5 sets. As a result, each person had 5 splits for each action shown in Table 3.2, and we merged each person's split to make a new split.

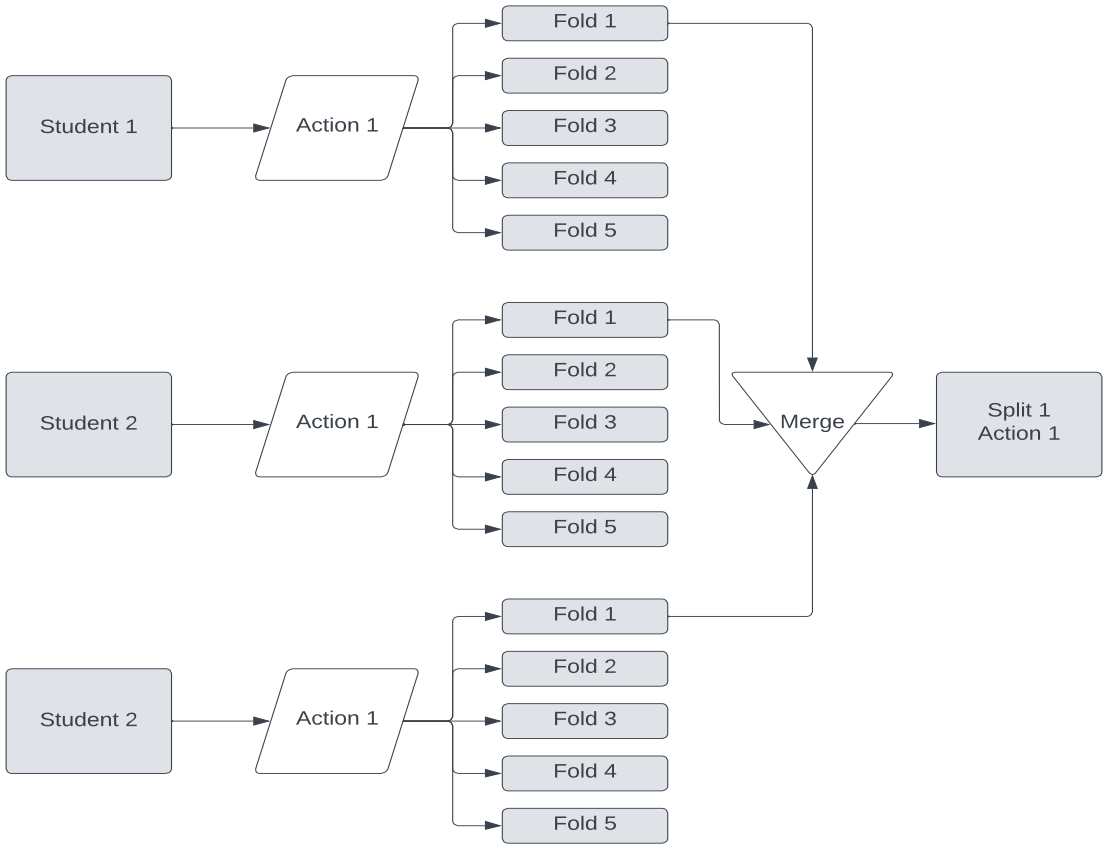

Figure 4.4: Group B Action split example

Figure 4.4 shows the action split process for group B. After we split the 10 actions and merged them, we had 5 splits, where each split contained one fifth of the number of the 10 actions from each person. At last, we did 4.1 to get 5 sets of text files.

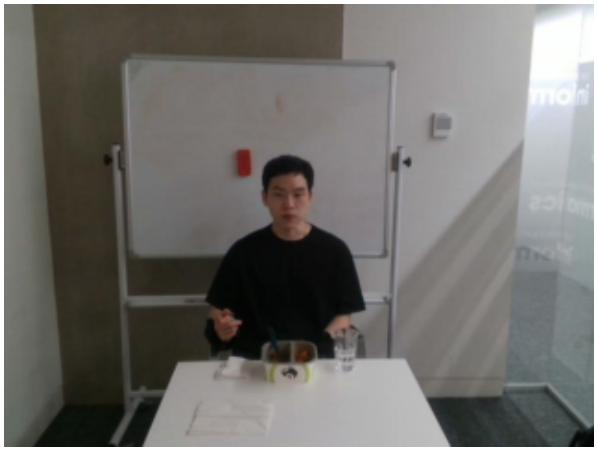

Figure 4.5: Original image

Figure 4.6: Normalization

Figure 4.7: Random cropping

Figure 4.8: Flipping

Figure 4.9: Color jittering

4.1.4 Data Augmentation

Now the dataset was prepared for training and testing the model, but we still needed to do a few more things to improve the process, such as data augmentation. There are several reasons why data augmentation is needed for deep learning - in brief, it could be said that it speeds up the process and improves the quality of the outcome.

Many researchers have tried to improve deep learning models in various ways, and augmentation is one of them. When AlexNet[22] was introduced it won many competitions, while defeating mainstream models such as Support Vector Machines(SVM) + Histogram of oriented Gradients(HoG). Data augmentation was heavily used for the first time in AlexNet. Naturally, other models adapted data augmentation, and VGGNet[23] was one of the models experimenting with data augmentation in many ways. The researchers working with this model found out that if the mean of RGB values is 0, losses are converged faster and the model has higher chances to escape from local optimums. Also, they resized images to make 3 different versions and cropped random areas of the resized images in 224*224 pixels. The VGGNet model could benefit from this by learning images by their parts. For example, when the model is learning dog images from resized images, the model could learn parts of dogs such as a head, a body, a tail, and etc. Then, if the model is classifying an image of a dog in a box only showing their tail, the model could classify the image as a dog because the model learnt that it is a dog from the resized images. Flipping and changing colors would bring similar benefits. VGGNet could decrease top-1 validation error around 3.4% by using these methods.

Similar to VGGNet, TPN also applies data augmentation to datasets. Figures 4.5 through Figure 4.9 show what data augmentations TPN applied to the dataset during the training phase. Note that the images were randomly cropped and resized by 224 by 224 pixels. Flipping was applied in a probability of 50%. TPN used slightly different data augmentation in testing. It resized images up to 256 by 256 pixels and cropped the center of images with the size of 224 by 224 pixels.

4.1.5 Summary

Finally our data preprocessing was done. We used the CSV files to extract the data that were necessary for making the Rawframe dataset needed to train the model. We used the frame and the action number data to create the annotations that indicate where the image folder is, how many images are in the folder, and what action number the images represent. We read the compressed files that had images and depth files and extracted certain images by using frame data we got from the CSV files, and saved the extracted images to the corresponding folder. After that, we split the whole Rawframe dataset into 5 sets by using the python script for 5-fold Cross-Validation.

Also, we applied data preprocessing and augmentation to the images - such as normalization, resizing, cropping, flipping, and color jittering. These allowed the model to train faster, avoid the gradient vanishing problem, and learn many aspects of the images, so the model could learn images faster and better.

Since we did 3 different experiments, the annotation files also needed to be adjusted for each experiment. We made training, validation, and testing text annotation files for each experiment. More details will be discussed in 5.3.1.

4.2 Implementation of TPN

4.2.1 Deep Learning Framework

Figure 4.10: Deep learning trends[24]

For deep learning, there are several famous frameworks such as TensorFlow and PyTorch. They have some pros and cons. TensorFlow is backed up by Google and has been used for close to a decade, so it is easier to find references to use. However, it is a quite complicated tool. In the case of PyTorch, unlike TensorFlow, it is not necessary to make a model from scratch. Figure 4.10 shows how the often keywords 'PyTorch' and 'TensorFlow' have been searched in 5 years. The graph shows that TensorFlow was more frequently searched, and that the number of searches has been decreasing, while PyTorch has become more popular than TensorFlow these days. We assume PyTorch has gained more attention than TensorFlow, because PyTorch seems to be more accessible than TesorFlow in terms of usability. Also, we already had experience in PyTorch, since many machine learning courses we took used PyTorch. Therefore, it seemed reasonable to choose a PyTorch-based tool - MMAction2.

MMAction2 has an application programming interface(API) to support the creation of datasets, train and test models, so we did not need to write a whole python code using a PyTorch library. It also contains action recognition algorithms, including TPN, so it was not necessary to implement TPN either. We cloned the MMAction2 GitHub

repository and installed dependencies and requirements including PyTorch. With the exception of configurations, the process was complete.

4.2.2 Configuration

Even though MMAction2 did most of the hard, tedious work for us, we still needed to write our own code for configurations. Fortunately, MMAction2 provides a configuration interface as well, so we can set everything up by importing its module named 'Config', so we do not need to edit each file for training and testing the model. The configurations could be categorized as data, training, and evaluation.

For the data processing configuration, the main concern is how to make the dataset fit the algorithm and how to change them to improve a model. There are several techniques we mentioned in 4.1.4. However, optimized configuration settings were already done, so we did not change anything there. Therefore, we only needed to establish which algorithm to use and where the training, testing datasets and their annotation files were placed.

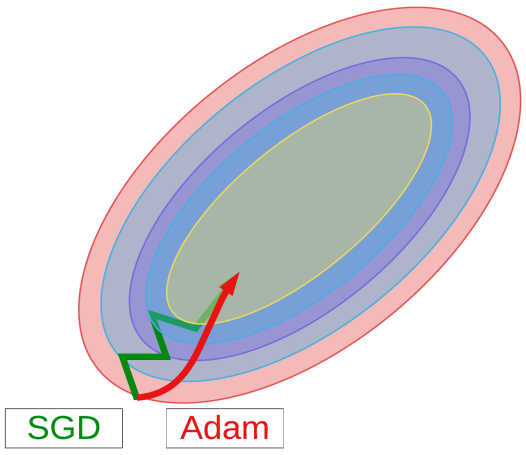

Figure 4.11: Visualization of Optimizers

Training configuration consisted of learning rate(LR), epoch, and optimizer. We set the LR between 1e-2 to 1e-5, epoch between 100 to 200, and the optimizer as either Stochastic Gradient Decent(SGD) or ADAptive Moment estimation(Adam). Since SGD focuses on speed and does not care about momentum, it is quite sensitive to LR. Adam seems to have solved this problem well, by combining step size and momentum, so Professor Fisher suggested we use it. In the TPN paper, they used between 0.01 to 0.0001 LR depending on epochs, up to 150 epochs, and SGD optimizer[17]. Similar to TPN, we found out that most action recognition algorithms including TIN and OmniSource used SGD for optimizer. We will discuss this further in Chapter 5.

$$\text{class accuracy} = \frac{\text{class hit}}{\text{class count}} \tag{4.1}$$

$$A = \{x_n \mid n \subseteq N, \quad 1 \leq n \leq 10, \quad x_n \text{ is a class accuracy of class n}\} \tag{4.2}$$

$$\text{mean class accuracy} = \overline{A} \tag{4.3}$$

For evaluation configuration, we could pick which data to store or show. We stored pth files, which contain data of the trained model, every 5 epochs, so we could use them later for testing at different epochs. Usually, top 1 and top 5 accuracy is used for showing a model's performance, but we decided to use top 1 and top 2 accuracy, since we only have 10 classes to classify. Top 5 accuracy would not be useful because it contains 50% of classes, so top 5 accuracy will be close to 90% most time, if the model is well trained. We also got mean class accuracy shown above, so we could conclude whether the model guessed all classes well or not.

4.2.3 TPN Structure

Unlike other algorithms, which capture visual tempos at the input-level, TPN does the same at the feature level. For example, the other algorithms make multiple videos with different frame rates by sampling an original video - this is computationally expensive. On the other hand, TPN only uses one video and captures visual tempos by leveraging the feature hierarchy, and it only needs the original video as input. As a result, TPN has M hierarchical features depending on M kinds of different frame rates.

To collect the features from a backbone network, TPN chooses a feature at some depth, and samples M different features with M different rates. The base feature $F_n = C \times T \times W \times H$ where C is channel[25], T is temporal, W is width, and H is height sizes for the feature. Therefore, the structure looks like a pyramid with $\{C \times \frac{T}{r_1} \times W \times H, ..., C \times \frac{T}{r_M} \times W \times H\}$ with M different rates $\{r_1, ..., r_M; r_1 < r_2 < ... < r_M\}$[17]. This looks similar to the structure shown in [25][26].

With these features shown at the backbone stage in Figure 2.2, TPN adds on an auxiliary classification head to align spatial features in the pyramid. We calculate the loss by using the loss function:

$$L_{total} = L_{CE,o} + \sum_{i=1}^{M-1} \lambda_i L_{CE,i} \tag{4.4}$$

where $L_{CE,o}$ is the original Cross-Entropy loss, $L_{CE,i}$ is the loss for i-th auxiliary head, and λ_i are balancing coefficients[17].

Stage	Layer	Output size
raw	-	8 ×224 ×224
conv_1	1×7×7, 64, stride 1, 2, 2	8×112×112
pool_1	1×3×3 max, stride 1, 2, 2	8×56×56
res_2	$\begin{bmatrix} 1\times1\times1, 64 \\ 1\times3\times3, 64 \\ 1\times1\times1, 256 \end{bmatrix} \times3$	8×56×56
res_3	$\begin{bmatrix} 1\times1\times1, 128 \\ 1\times3\times3, 128 \\ 1\times1\times1, 512 \end{bmatrix} \times4$	8×28×28
res_4	$\begin{bmatrix} 3\times1\times1, 256 \\ 1\times3\times3, 256 \\ 1\times1\times1, 1024 \end{bmatrix} \times6$	8×14×14
res_5	$\begin{bmatrix} 3\times1\times1, 512 \\ 1\times3\times3, 512 \\ 1\times1\times1, 2048 \end{bmatrix} \times3$	8×7×7
global average pool, fc		1×1×1

Figure 4.12: TPN backbone (Figure copied from [17])

TPN has the 3D ResNet-50 back bone structure introduced in [25] where it learns spatial and temporal features. TPN gets spatial features from res_2 to res_5, and obtains temporal features from a convolutional layer and max-pooling layer. TPN aggregates them and makes the final predictions.

4.2.4 Summary

For the sake of convenience, we used MMAction2 for our experiments. MMAction2 has useful tools, so there were few things to do, such as preparing the Rawframe dataset and 5-fold cross-validation. After these, we set the configurations for TPN such as epochs, LR, and optimizers. We also quickly checked how TPN would work with the training as well.

Chapter 5

Evaluation

The main goal for this project is recognising eating actions, consisting of sub-actions. However, sub-actions might happen for a very short time and could be misclassified if the network cannot understand the context of the sub-actions. For example, eating and drinking share almost the same joint movements, but they are different actions. TPN uses visual tempos to try to solve these problems; e.g. arm movement for drinking might be slower than eating and and the arm/hand might spend more time around the mouth. We wanted to know if TPN works well and could be used for the next step - identifying motor deterioration in various circumstances. We experimented with three different situations to see results such as accuracy, generalisation, and improvement.

- Experiment 1 was meant to answer the question "how well can my network classify actions?". We used the dataset from both group A and B for training and testing, to see the network's performance with well-prepared data.
- Experiment 2 adressed the question "how well can my network generalise its classsifcation?". We used all of group A's data for training and group B's data for testing.
- Experiment 3 aimed to see "how well can my network perform with a little training data?". We separated group B's data into 2 parts and used one part for training and the other for testing.

We will discuss how we split the data for each experiment, and evaluate the experiments in the corresponding sections.

5.1 SGD and Adam Optimizer Comparison

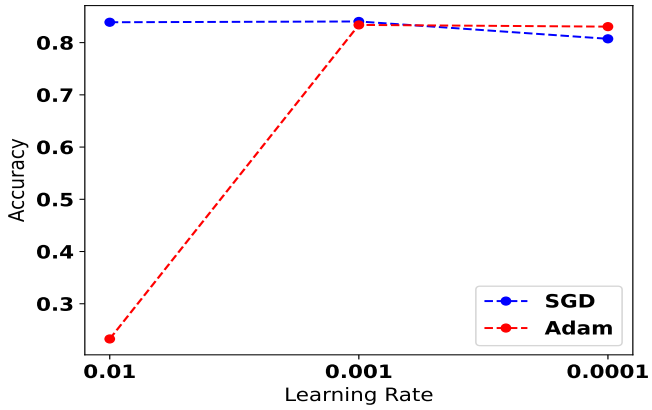

Figure 5.1: Experiment 1 SGD and Adam optimizer accuracy comparison

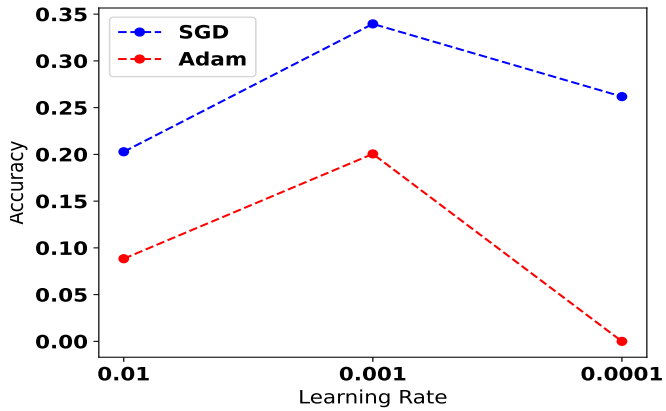

Figure 5.2: Experiment 2 SGD and Adam optimizer accuracy comparison

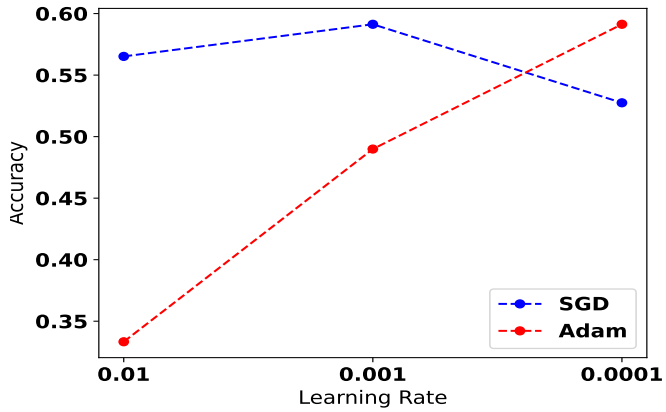

Figure 5.3: Experiment 3 SGD and Adam optimizer accuracy comparison

At first, we compared SGD and Adam optimizer for the further tests. We planned to use the better optimizer out of the two tools. Since Adam is known to be the best optimizer, we expected accuracy with Adam would be higher, but the result was not the same as we expected. We picked the highest accuracy between 25 to 100 epochs for different LRs for SGD and Adam to compare. Figure 5.1, 5.2, and 5.3 shows accuracy with different LRs from 0.01 to 0.0001 for SGD and Adam optimizer in split 1. As the figures show, SGD has higher accuracy in all LRs in general. The more interesting point was that Adam seems to be more unstable than SGD, which is known to show variant accuracy depending on LR. Figure 5.1 shows that Adam got close to 0 accuracy at 0.01 LR compared to SGD, which got 0.6. Since experiment 1 has the full dataset, we wondered how Adam would work in conditions in which the data are not sufficient. In experiment 2, the network was tested with the data which did not appear in the training phase. It seems SGD adjusts better to these conditions than Adam. In experiment 3, a small amount of data was used for training, while similar data to the training data were used for testing. Adam was still unstable in lower LR, but accuracy kept increasing as LR increased, unlike in the case of SGD, where accuracy declined at 0.0001 LR. Overall, SGD seems to be a better choice, except for the case where we have insufficient training data to test data similar to that which was used in training.

We tried to investigate why SGD performed better than Adam in action recognition, as opposed to other fields. such as image classification, etc., so we were looking for any evidence as to why the researchers of TPN, TIN, and OmniSource chose SGD instead of Adam. We only found that they mentioned they followed the common practice. Other previous action recognition algorithms, such as C3D[27] and TSN[28], which were referred to as related work to TPN, also used SGD without stating a clear reason.

Because of the result we found, we decided to use SGD instead of Adam optimizer for the experiments.

5.2 Training Method

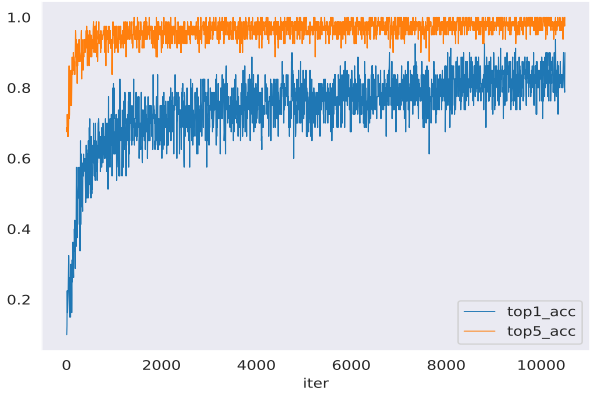

Figure 5.4: Training accuracy

In most cases of training, Top-1 accuracy reached the peak in 50 epochs as shown in Figure 5.4. It looks like the network got overtrained after a few epochs, but Top-1 test accuracy increased after those few epochs. Therefore, it was not enough to check whether the network was trained enough or not by looking at training accuracy. However, it was also not sufficient to check the Top-1 test accuracy, since this would also not give the full picture.

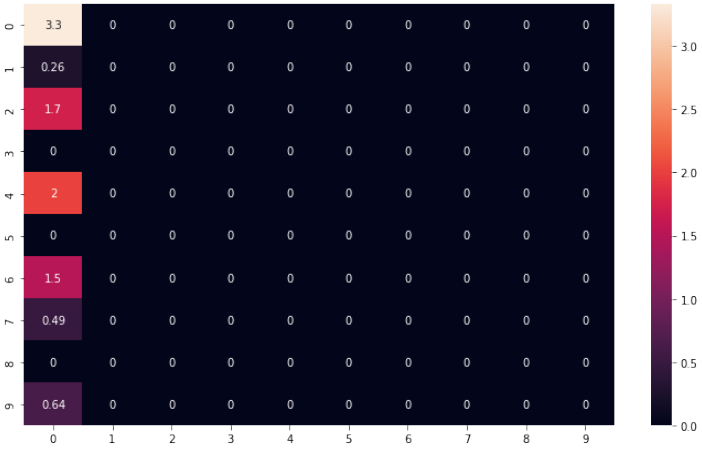

Figure 5.5: Confusion matrix - bad class accuracy

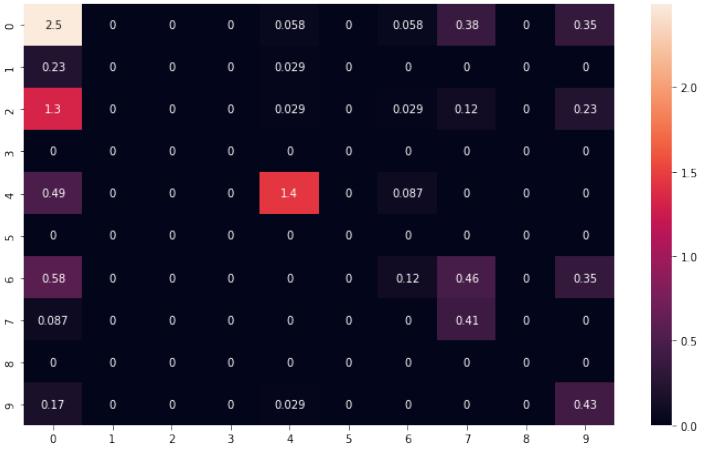

Figure 5.6: Confusion matrix - better class accuracy

Figure 5.5 and 5.6 are confusion matrices with different epochs, and they have similar Top-1 test accuracy. However, Figure 5.5 could get accuracy similar to that in Figure 5.6 only because the data mostly consisted of the action 'other' and it misclassified all classes as 'other'. Figure 5.5 could not classify classes except the action 'other', while Figure 5.6 could hit other classes. Therefore, we could not rely only on Top-1 test accuracy, so we also got mean class accuracy, which was the mean of each class's accuracy. As a result, considering both Top-1 and mean class accuracy seemed to be better to get the best test results.

We tried to run different tests to get a baseline. For example, we set LR from 1e-1 to 1e-5 and epoch from 50 to 200 for the first split. Then LR was adjusted from 1e-2 to 1e-4, and epoch was set from 100 to 150, because there were no significant improvements in accuracy in low or high LRs and epochs. We obtained Top-1, Top-2, and mean class accuracies during the tests. We focused on Top-1 accuracy and mean class accuracy, since they represent the performance of the network and how well each class is classified. After the tests, mean Top-1 accuracy and STandard DEViation(STDEV) were obtained to see the performance in general, and the test results for each split were similar to each other. Table 5.2, 5.4, and 5.6 show the result of the test. they consist of the best overall test result, the best test result for each split, and mean and STDEV of the best test result.

5.3 Experiment 1

5.3.1 Data Split

Group	Data
A	A1, A2, A3, A4, A5
B	B1, B2, B3, B4, B5

Split	Train	Val	Test
1	A3, A4, A5, B3, B4, B5	A1, B1	A2, B2
2	A4, A5, A1, B4, B5, B1	A2, B2	A3, B3
3	A5, A1, A2, B5, B1, B2	A3, B3	A4, B4
4	A1, A2, A3, B1, B2, B3	A4, B4	A5, B5
5	A2, A3, A4, B2, B3, B4	A5, B5	A1, B1

Table 5.1: Experiments 1 data split

To test the accuracy of our network, we used all data we obtained to get the best output. Table 5.1 shows how the data split into train, validation, and test sets. First, we used 5-fold cross-validation, which we mentioned in 4.1.3, to get 5 folds of group A and B. Second, we used 3 folds from each group A and group B for training, 1 fold from each group for validation, and also 1 split from both groups for testing. We used the same method mentioned in Table 4.1 to make 5 splits for the experiment.

5.3.2 Experiment Description

This experiment was for checking the best accuracy which the network could get from the current dataset. We used all data from group A and B, and did 5-fold cross validation to get the best hyperparameters.

Best overall					
Split	LR	Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
3	1e-3	50	91.17%	96.67%	88.38%

Best Test Accuracy			
Split	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
1	84.05%	95.35%	81.24%
2	85.22%	94.91%	81.87%
3	91.17%	96.67%	88.38%
4	85.25%	95.03%	84.06%
5	85.74%	95.35%	84.49%

	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
Mean	86.29%	95.46%	84.01%
STDEV	2.28	0.70	2.8

Table 5.2: Experiments 1 result

5.3.3 Experiment Result

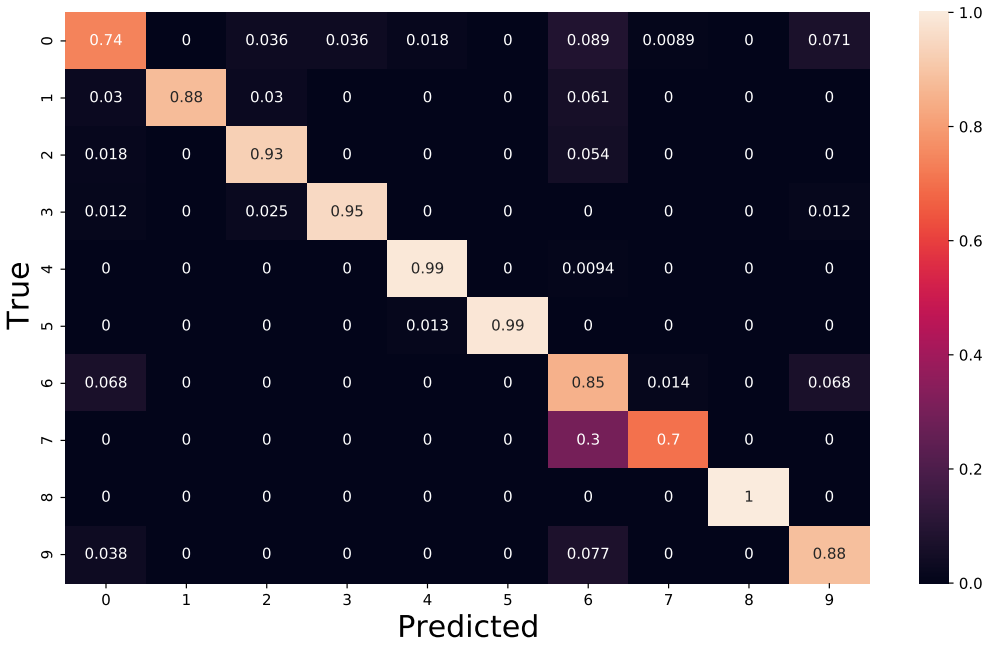

Figure 5.7: Experiment 1 best confusion matrix

The result shows that the accuracy was reasonable. The best accuracy we got was 91.17% with 50 epochs. There was not much difference between Top-1 and mean class accuracy, so it seems the network was well trained. STDEV also showed that

there were not many differences among splits. Interesting point here is that 30% of the action number 7 'no action' was misclassified as the action number 6 'chewing'. Confusion matrix Figure 5.7 shows it clearly. We assume that either TPN is not able to differenciate the motions without much movement or the dataset has not enough data since the count of 'No action' class is only 53 while the count of 'chewing' is 359. Also, action number 8 'Pick up food from a container with tools in both hands' got 100% accuracy. We assume that this is because the time stamps of the action are slightly longer than other actions, so the action has more total frames than other actions. Or, the hyperparameters were well tuned for the action since the action was not better than other actions in other hyperparameters.

Overall, the result shows that TPN would work well if there were enough training data for someone whose eating action we want to recognize, and if we do hyperparameter tuning.

5.4 Experiment 2

5.4.1 Data Split

Group	Data
A	A1, A2, A3, A4, A5
B	B1, B2, B3, B4, B5

Split	Train	Val	Test
1	A2, A2, A3, A4	A1	B1, B2, B3, B4, B5
2	A3, A4, A5, A1	A2	B1, B2, B3, B4, B5
3	A3, A4, A5, A1	A3	B1, B2, B3, B4, B5
4	A4, A5, A1, A2	A4	B1, B2, B3, B4, B5
5	A5, A1, A2, A3	A5	B1, B2, B3, B4, B5

Table 5.3: Experiments 2 data split

Similar to Experiment 1, we used 5-fold cross validation to get 5 splits from each group, A and B. However, this time we only used the group A data for training and made the group B data a testing set. 4 splits and 1 split from group A were used for training and validation, respectively, and all splits from group B were dedicated to testing as shown in Table 5.3.

5.4.2 Experiment Description

Here we wanted to know if the model classifies unknown people's actions well. The network was trained only with group A, so it saw group B only in testing. People have different eating motions, so this experiment would reveal whether the network could recognize the same actions with different motions. We also tried various hyperparameters at the beginning, and fixed them after we found a boundary of overfitting, where Top-1 and mean accuracy started decaying.

Best overall					
Split	LR	Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
4	1e-2	100	38.09%	52.36%	13.71%

Best Test Accuracy			
Split	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
1	33.96%	46.93%	13.92%
2	34.91%	48.70%	12.81%
3	32.85%	49.29%	12.17%
4	38.09%	52.36%	13.71%
5	35.73%	53.77%	13.62%

	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
Mean	35.71%	50.21%	13.25%
STDEV	1.53	2.79	0.73

Table 5.4: Experiments 2 result

5.4.3 Experiment Result

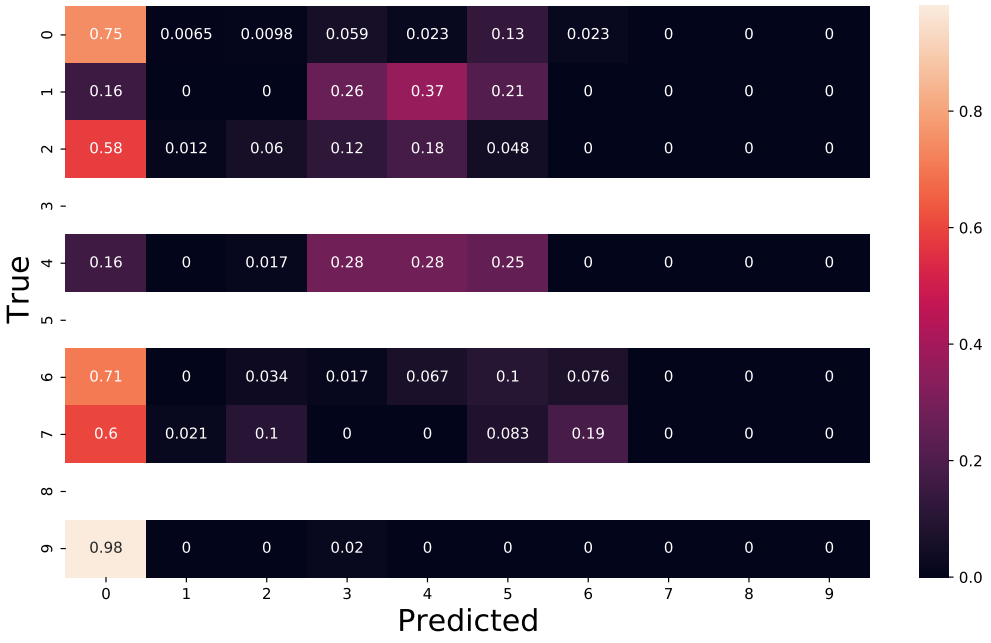

Figure 5.8: Experiment 2 best confusion matrix

Table 5.4 shows the summary of Experiment 2. The data look to be well split, as we can see the low numbers of STDEV, but considering the accuracy we got, the network does not seem good for generalising its classfication. Top-1 accuracy could not go above

40%, and moreover, mean class accuracy was around 13% for all LRs. This means the network could not classify some actions at all, and this is shown as the confusion matrix in Figure 5.8. Even though it was made from the best accuracy in Experiment 2, the network failed to classify most actions except 0. Since group B did not have actions 3, 5, and 8, corresponding rows remained blank. The network classified actions mostly as 'other' except 'pick up food from a container with both hands' and 'Eat it'. Even the two exceptional actions was misclassified as other actions. Therefore, it seems the network found out that most actions in group B are more similar to 'other' in group A than other actions in group A. This might be because people have quite different eating behavior with other.

5.5 Experiment 3

5.5.1 Data Split

Name	Sequence	Data
Minsung	1	M1A, M1B, M1C, M1D
	2	M2
Fadl	1	F1A, F1B, F1C, F1D
	2	F2
Priya	1	P1A, P1B, P1C, P1D
	2	P2

Experiment	Train	Val	Test
3	M1A, M1B, M1C, F1A, F1B, F1C, P1A, P1B, P1C	M1D, F1D, P1D	M2, F2, P2

Table 5.5: Experiments 3 data split

Unlike in the other 2 experiments, Experiment 3 had the group B dataset only. We folded the first sequence of each student in group B and made 4 splits. The second sequence was used as a whole. Then we did the 4-fold cross-validation for the first sequences for training, and used the second sequences for testing.

5.5.2 Experiment Description

We discovered that the network did not work well without the training data from the previous experiment, bleonging to someone, whose actions we wanted to recognise. In this experiment, we wanted to figure out if the model could improve performance to be better than in Experiment 2 by adding a little amount of data from that inidividual. This experiment is similar to Experiment 1, but we only use limited data for training. This might prove that the network cannot differentiate between the same actions done by different people, so it would be better to have a small amount of related data rather than a large amount of unrelated data.

Best overall					
Split	LR	Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
4	1e-2	75	62.03%	76.52%	52.03%

Best Test Accuracy			
Split	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
1	59.13%	73.33%	51.15%
2	61.16%	74.49%	50.71%
3	60.29%	73.91%	56.14%
4	62.03%	76.52%	52.03%

	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
Mean	60.65%	74.56%	52.51%
STDEV	1.24	1.39	2.48

Table 5.6: Experiments 3 result

5.5.3 Experiment Result

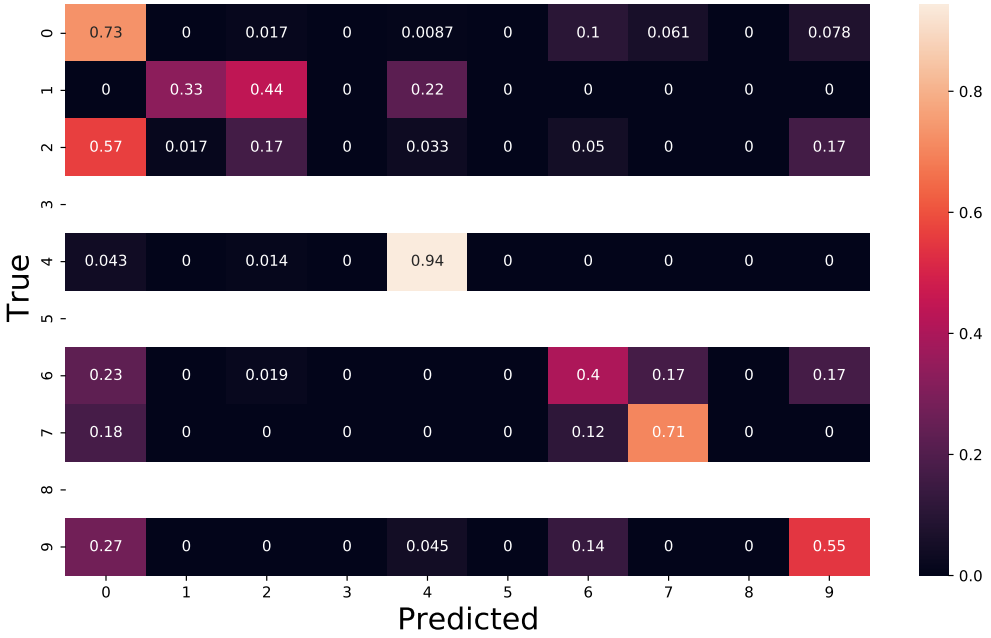

Figure 5.9: Experiment 3 best confusion matrix

The network showed its improvement clearly by Top-1 and mean class accuracy. Top-1 accuracy values were at least 1.5 times better and mean class accuracy values were almost 3 times to 4 times better than Experiment 2's values. Still, many actions were misclassified as 'other', but the confusion matrix in Figure 5.9 shows a more diverse

Mean Class Accuracy			
Split	TPN	TIN	OmniSource
1	13.92%	9.10%	14.84%
2	12.81%	7.03%	14.19%
3	12.17%	6.98%	8.11%
4	13.71%	6.89%	15.78%
5	13.62%	5.85%	13.43%

Table 5.7: Experiments 2 mean class accuracy for all algorithms

Top-1 Accuracy		
	TPN	OmniSource
Mean	60.65%	55.83%
STDEV	1.24	3.83

Mean Class Accuracy		
	TPN	OmniSource
Mean	52.51%	42.74%
STDEV	2.48	5.57

Table 5.8: Experiments 3 result for all algorithms

shape than in Experiment 2. While the accuracy of most actions were below 50%, 'eat it' and 'no action got 94% and 71% accuracy. Especially group B only has 48 'no action' actions, but still it managed to get high accuracy. We assume that this is because it was relatives easy task since there was no movement in the action. Other tests support this assumption that 'no action' accuracy was above 50% if epoch was larger than 25. We found out that the mean accuracy with 0.01 LR was higher than with 0.001 LR with 100 epochs. We assumed the network was not fully trained, so we trained the network with 200 epochs for 0.0001 LR, but we could not see any improvement.

5.6 Algorithm comparison

Overall, TPN, TIN, and OmniSource have similar accuracy in all experiments. The only difference is that TIN has quite low mean class accuracy in experiment 2. Table 5.7 shows the comparison. We guess TIN loses certain information while it make 2D information to 1D information. In experiment 3, TPN looks better than OmniSoruce in all aspect

TPN seems works better with few dataset than OmniSource. TPN is more stable than OmniSource if we consider STDEV.

Chapter 6

Summary

6.1 Discussion

In this paper, we worked through sub-problems to achieve the project goal.

- TPN review: Action recognition algorithms such as C3D[27] and TSN[28] were reviewed to understand TPN. There were many new concepts for us; for example, 3D backbone and visual tempo modelling. Since spatio-temporal action recognition is a relatively new concept, it was hard to find information about the algorithms except in research papers.

- Labelling: The videos were labelled by using the annotation tool VIA. It was a manual task, so a few instances of mislabelling happened, which we needed to remove in data preprocessing. Also, there was no clear standard to classify actions. Since people do not have the same perspective to define actions, TPN would show different accuracy depending on the person who labelled the actions.

- Cross validation: 4-fold and 5-fold cross validation were done to avoid overfitting and find the best parameters. They was done by a python code which we mentioned in 4.1.3.

- Data preprocessing: The data were processed for training and testing to be used by TPN. We removed the irrelevant data and used the data augmentation method to enrich the data. We used the PyTorch library to do this. We did not manipulate data augmentation settings, such as normalization values, size of the images, etc.

- Experiments: We did the 3 experiments by using the data and created an accuracy table and a confusion matrix for each test. The result shows TPN's performance in the 3 conditions. This could give us a hint for the next project's setting. Most difficulties here happened due to the unstable Colab environment in which sessions were disconnected after certain time, while training the model.

6.2 Plan for Part 2

- Since we only used RGB data in this project, we might use depth data next time. Also, we need to find action recognition algorithms using depth data.

- We need to learn action recognition algorithms further. For example, we might investigate why most action recognition algorithms use SGD optimizer than Adam optimizer.
- After we get sufficient accuracy on eating action recognition, we might gather more videos in which people are wearing weight wrists for motor deterioration detection.

6.3 Conclusion

Since population ageing has become an issue over the years, the lack of medical staff has also proven to be a significant challenge. In this paper, we suggest a marker-less camera based motion estimation system to monitor elders' everyday life to diagnose any severe health problem. The elders would have a higher quality of life thanks to using the system, since they could be freed from regular medical checkups.

We tried to establish the system by using action recognition algorithms, and here we used TPN. We found the best hyperparameters for TPN through cross validation and data preprocessing. Also, we figured out TPN works well if there is enough data for the target - the person whose actions we want to recognise. However, it does not classify well if there are no data for the target in the training phase. The good news is that TPN performs relatively well with little data of the target.

There were several questions remaining. The quality of labels is highly dependent on individuals, and it could be only improved through cross checking by other people which is also a time consuming task. We could not do this due to the lack of time. The other problem was about trade-off. It seems TPN learns fast, so it achieves reasonable accuracy around 50 epochs similar to accuracy around 100 or more epochs. It was not assured to sacrifice accuracy for efficiency, since TPN is required to be tuned to individual data according to Experiment 2, which demands a lot of resources. This needs a certain level of efficiency to be dealt with, while the project is about health, which requires peak accuracy.

To address these problems, we might try the following methods. For the label problem, we might create a checker. For example, everyone participating in the project labels the same video, and the checker will check each person's action labels and their time stamps. If any action or time stamp was noticeably different than others, the action and the time stamp will be manually checked by people for standardization. For the trade-off problem, we might try to use the meta learning[29] method. It would allow us to avoid training the network every time we get new data.

Appendix A

Experiment Results

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.7774	0.8887	0.7380
50	0.7691	0.8738	0.6732
75	0.8389	0.9419	0.8360
100	0.8372	0.9385	0.8047

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.7890	0.9236	0.7361
50	0.8239	0.9402	0.7893
75	0.8306	0.9369	0.8226
100	0.8405	0.9535	0.8124
125	0.8505	0.9502	0.8343
150	0.8405	0.9535	0.8077

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5930	0.8156	0.4330
50	0.7625	0.9003	0.6874
75	0.7924	0.9053	0.7653
100	0.8073	0.9103	0.7619
125	0.8056	0.9169	0.7628
150	0.8123	0.9153	0.7644
175	0.8040	0.9169	0.7570
200	0.8140	0.9203	0.7658

Table A.1: Experiment 1: Split 1 Optimizer SGD result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.1279	0.2791	0.1000
50	0.2326	0.4186	0.1462
75	0.2076	0.3920	0.1229
100	0.2226	0.4169	0.1377

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8073	0.9219	0.7521
50	0.8256	0.9153	0.7641
75	0.8223	0.9169	0.8203
100	0.8339	0.9385	0.8152

Adam Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8040	0.9203	0.7184
50	0.8306	0.9203	0.7990
75	0.8156	0.9219	0.7918

Table A.2: Experiment 1: Split 1 Optimizer Adam result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8243	0.9327	0.8403
50	0.8046	0.9015	0.7361
75	0.8489	0.9376	0.7781
100	0.8539	0.9589	0.7937

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8407	0.9343	0.8044
50	0.8456	0.9327	0.8143
75	0.8391	0.9409	0.7805
100	0.8522	0.9491	0.8187
125	0.8456	0.9376	0.8191
150	0.8506	0.9442	0.8426

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.6092	0.8144	0.4352
50	0.7734	0.9179	0.7153
75	0.8128	0.9179	0.7591
100	0.8210	0.9343	0.7604
125	0.8144	0.9310	0.7461
150	0.8194	0.9327	0.7464
175	0.8194	0.9261	0.7526
200	0.8194	0.9294	0.7498

Table A.3: Experiment 1: Split 2 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8383	0.9317	0.7498
50	0.8867	0.9600	0.8140
75	0.875	0.9367	0.8346
100	0.9083	0.9667	0.8925

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8683	0.9550	0.8177
50	0.9117	0.9667	0.8838
75	0.8750	0.9600	0.8667
100	0.8983	0.9600	0.8831
125	0.8917	0.9617	0.8764
150	0.8933	0.9667	0.8715

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.6100	0.8033	0.4626
50	0.7633	0.9033	0.6824
75	0.8633	0.9533	0.7807
100	0.8567	0.9550	0.7770
125	0.8667	0.9517	0.7935
150	0.8800	0.9517	0.8365
175	0.8800	0.9533	0.8495
200	0.8767	0.9533	0.8258

Table A.4: Experiment 1: Split 3 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.7993	0.9348	0.8110
50	0.8165	0.9348	0.7376
75	0.8302	0.9297	0.7398
100	0.8525	0.9383	0.8167

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.8130	0.8988	0.8195
50	0.8336	0.9417	0.8154
75	0.8174	0.9537	0.8613
100	0.8525	0.9503	0.8406
125	0.8611	0.9554	0.8581
150	0.8542	0.9537	0.8491

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.6346	0.8130	0.4797
50	0.7770	0.8988	0.6888
75	0.8319	0.9245	0.7489
100	0.8233	0.9211	0.7367
125	0.8319	0.9331	0.7446
150	0.8422	0.9262	0.7566
175	0.8285	0.9262	0.7347
200	0.8405	0.9280	0.7567

Table A.5: Experiment 1: Split 4 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.7430	0.8922	0.7425
50	0.8292	0.9303	0.7692
75	0.7745	0.9088	0.6892
100	0.8557	0.9486	0.8270

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.7944	0.9270	0.8267
50	0.8574	0.9353	0.8449
75	0.8408	0.9536	0.8394
100	0.8507	0.9502	0.8428
125	0.8574	0.9420	0.8530
150	0.8507	0.9536	0.8383

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.6364	0.8165	0.4810
50	0.7736	0.8971	0.6864
75	0.8302	0.9228	0.7492
100	0.8250	0.9245	0.7403
125	0.8319	0.9314	0.7564
150	0.8370	0.9228	0.7517
175	0.8353	0.9365	0.7450
200	0.8422	0.9262	0.7606

Table A.6: Experiment 1: Split 5 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.0884	0.6863	0.0305
50	0.2028	0.5849	0.1429
75	0.0000	0.5271	0.0000
100	0.0000	0.6639	0.0000

SGD Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2583	0.3679	0.1088
50	0.2913	0.4104	0.1273
75	0.2583	0.4127	0.1001
100	0.3396	0.4693	0.1392

Adam Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.0000	0.0000	0.0000
50	0.2005	0.3467	0.1109
75	0.1356	0.2901	0.0828
100	0.1439	0.3255	0.0723

SGD Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.2028	0.2323	0.0862
50	0.2394	0.2913	0.0871
75	0.2453	0.3373	0.0909
100	0.2618	0.4021	0.1003

Adam Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.0000	0.0000	0.0000
50	0.0000	0.0000	0.0000

Learning Rate = 1e-5			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.0024	0.6521	0.0015
50	0.0389	0.6757	0.0240
75	0.1038	0.6639	0.0640
100	0.0000	0.6722	0.0000
200	0.0861	0.6651	0.0531

Table A.7: Experiment 2: Split 1 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2689	0.4941	0.1026
50	0.2347	0.4245	0.0842
75	0.3113	0.4941	0.0953
100	0.3396	0.4705	0.1173

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.3491	0.4870	0.1281
50	0.2818	0.4245	0.1288
75	0.3208	0.4175	0.1170
100	0.3290	0.4564	0.1195

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.2642	0.4104	0.0964
50	0.2547	0.4599	0.1007
75	0.3149	0.4186	0.1238
100	0.3314	0.4210	0.1180

Learning Rate = 1e-5			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.0035	0.6686	0.0019
50	0.0483	0.6816	0.0265
75	0.0932	0.6675	0.0510
100	0.1108	0.6675	0.0607

Table A.8: Experiment 2: Split 2 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2229	0.4257	0.1246
50	0.2370	0.4599	0.0776
75	0.2771	0.4410	0.1000
100	0.2983	0.4811	0.1060

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2748	0.4316	0.0988
50	0.3432	0.4870	0.1363
75	0.3243	0.4057	0.1230
100	0.3455	0.4233	0.1232

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2771	0.4729	0.1116
50	0.2854	0.5189	0.1333
75	0.3526	0.4575	0.1196
100	0.3585	0.4929	0.1217

Learning Rate = 1e-5			
Epoch	Top-1 Acc.	Top-5 Acc.	Mean Class Acc.
25	0.0094	0.6509	0.0052
50	0.0519	0.6639	0.0284
75	0.0837	0.6627	0.0459
100	0.0790	0.6698	0.0433
150	0.1203	0.6639	0.0741
200	0.0908	0.6627	0.0497

Table A.9: Experiment 2: Split 3 result

Learning Rate = 1e-01			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
10	0.0000	0.0000	0.0000
20	0.0000	0.0000	0.0000
30	0.2028	0.2028	0.1429

Learning Rate = 1e-02			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.1981	0.3691	0.1037
50	0.3384	0.5307	0.1214
75	0.3432	0.4882	0.1095
100	0.3809	0.5236	0.1371

Learning Rate = 1e-03			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2972	0.4682	0.1376
50	0.2382	0.4021	0.1197
75	0.3137	0.4634	0.1106
100	0.3243	0.4929	0.1162

Learning Rate = 1e-04			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2028	0.2323	0.0862
50	0.2394	0.2913	0.0871
75	0.2453	0.3373	0.0909
100	0.2618	0.4021	0.1003

Table A.10: Experiment 2: Split 4 result

Learning Rate = 1e-02			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.1486	0.4599	0.1149
50	0.2618	0.5083	0.0918
75	0.2252	0.5012	0.1041
100	0.3573	0.5377	0.1362

Learning Rate = 1e-03			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
20	0.2759	0.4658	0.1092
25	0.3078	0.4941	0.1167
75	0.3196	0.4976	0.1227
100	0.3184	0.4646	0.1166

Learning Rate = 1e-04			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.2276	0.2936	0.0936
50	0.2748	0.3927	0.1049
75	0.3031	0.4269	0.1158
100	0.2877	0.3868	0.1087

Table A.11: Experiment 2: Split 5 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.4725	0.6841	0.2946
50	0.3159	0.571	0.2827
75	0.5072	0.658	0.3724
100	0.5652	0.7072	0.4923

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5159	0.6986	0.4680
50	0.5304	0.7159	0.4712
75	0.5913	0.7333	0.5115
100	0.5536	0.7159	0.5070

Adam Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.4377	0.5304	0.2187
50	0.4812	0.6522	0.296
75	0.5101	0.6783	0.4307
100	0.5275	0.6783	0.4489

Table A.12: Experiment 3: Split 1 Optimizer SGD result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.3333	0.5362	0.1429
50	0.3333	0.5362	0.1429
75	0.3333	0.5362	0.1429
100	0.3333	0.5362	0.1429

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.3942	0.5449	0.1857
50	0.3942	0.5449	0.1857
75	0.4406	0.6174	0.2325
100	0.4899	0.6348	0.4349

Adam Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5884	0.7217	0.4742
50	0.5652	0.7217	0.4522
75	0.5913	0.7507	0.4342
100	0.5913	0.7536	0.5115

Table A.13: Experiment 3: Split 1 Optimizer Adam result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5014	0.6638	0.2944
50	0.5275	0.7304	0.458
75	0.5420	0.7623	0.4062
100	0.5768	0.7681	0.4329

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5391	0.7217	0.4178
50	0.5391	0.7188	0.4628
75	0.5043	0.7101	0.3156
100	0.6116	0.7449	0.5071

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.3884	0.5188	0.1816
50	0.4725	0.6203	0.3191
75	0.5072	0.6870	0.3398
100	0.5130	0.6899	0.3574

Table A.14: Experiment 3: Split 2 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.4406	0.6522	0.2466
50	0.5710	0.7362	0.4882
75	0.5304	0.7217	0.4534
100	0.6029	0.7391	0.5614

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.542	0.7014	0.5011
50	0.5449	0.7478	0.5159
75	0.5536	0.7130	0.4923
100	0.6000	0.7797	0.5505

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.4145	0.5217	0.2008
50	0.4696	0.6319	0.3123
75	0.5072	0.6725	0.3563
100	0.5246	0.6870	0.3673

Table A.15: Experiment 3: Split 3 result

Learning Rate = 1e-2			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5159	0.6899	0.3224
50	0.5072	0.6870	0.4139
75	0.6203	0.7652	0.5203
100	0.6145	0.7565	0.5448

Learning Rate = 1e-3			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.5275	0.7130	0.3654
50	0.5420	0.7275	0.5192
75	0.6058	0.7420	0.5399
100	0.5971	0.7420	0.5189

Learning Rate = 1e-4			
Epoch	Top-1 Acc.	Top-2 Acc.	Mean Class Acc.
25	0.4087	0.5246	0.1967
50	0.4725	0.6551	0.2873
75	0.4986	0.6812	0.3235
100	0.5246	0.6986	0.3704

Table A.16: Experiment 3: Split 4 result

Bibliography

[1] MMAction2, "Spatio-temporal action recognition." [Online]. Available: https://raw.githubusercontent.com/open-mmlab/mmaction2/master/resources/spatio-temporal-det.gif

[2] W. E. Forum, "How long will people live in the future?" [Online]. Available: https://www.weforum.org/agenda/2021/07/how-long-will-people-live-in-the-future/

[3] Pulse, "Korea's population shrinks for the first time in 2020 on record low birth rate." [Online]. Available: https://m.pulsenews.co.kr/view.php?year=2021&no=182191

[4] M. N. Today, "How does estrogen affect osteoporosis?" [Online]. Available: https://www.medicalnewstoday.com/articles/estrogen-and-osteoporosis

[5] D. Morgan, J. Reed, and A. Palmer, "Moving from hospital into a care home – the nurse's role in supporting older people," *Journal of Clinical Nursing*, vol. 6, no. 6, pp. 463–471, 2007.

[6] M. A. Raza, "Visual assessment of long-term changes of activity levels in elderly people," *Institute of Perception, Action and Behaviour, School of Informatics, University of Edinburgh*, 2021.

[7] K. Anderson, "Why changing eating habits (permanently) is so hard." [Online]. Available: https://www.psychologytoday.com/intl/blog/behernow/201709/why-changing-eating-habits-permanently-is-so-hard

[8] S. L. Colyer, M. Evans, D. P. Cosker, and A. I. T. Salo, "A review of the evolution of vision-based motion analysis and the integration of advanced computer vision methods towards developing a markerless system," *Sports Medicine*, 2018.

[9] K. Fan, P. Wang, and S. Zhuang, "Human fall detection using slow feature analysis," *Multimedia Tools and Applications*, vol. 78, no. 7, pp. 9101–9128, 2019.

[10] N. Barla, "A comprehensive guide to human pose estimation." [Online]. Available: https://www.v7labs.com/blog/human-pose-estimation-guide

[11] C. Zheng, W. Wu, T. Yang, S. Zhu, C. Chen, R. Liu, J. Shen, N. Kehtarnavaz, and M. Shah, "Deep learning-based human pose estimation: A survey," 2021.

[12] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[13] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, "Crowdpose: Efficient crowded scenes pose estimation and a new benchmark," *arXiv preprint arXiv:1812.00324*, 2018.

[14] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[15] M. Fiaz, A. Mahmood, S. Javed, and S. K. Jung, "Handcrafted and deep trackers: Recent visual object tracking approaches and trends," *ACM Computing Surveys*, vol. 52, no. 4, pp. 1–44, 2019.

[16] S. Ranasinghe, , F. A. Machot, and H. C. Mayr, "A review on applications of activity recognition systems with regard to performance and evaluation," *International Journal of Distributed Sensor Networks*, vol. 12, no. 8, 2016.

[17] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, "Temporal pyramid network for action recognition," *Conference on Computer Vision and Pattern Recognition*, 2020.

[18] H. Shao, S. Qian, and Y. Liu, "Temporal interlacing network," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, pp. 11 966–11 973, 2020.

[19] H. Duan, Y. Zhao, Y. Xiong, W. Liu, and D. Lin, "Omni-sourced webly-supervised learning for video recognition," *arXiv preprint arXiv:2003.13042*, 2020.

[20] H. Pirsiavash and D. Ramanan, "Parsing videos of actions with segmental grammars," *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[21] A. G. Abhishek Dutta and A. Zisserman, "Vgg image annotator (via)." [Online]. Available: https://www.robots.ox.ac.uk/~vgg/software/via/

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *arXiv preprint arXiv:1409.1556*, vol. 1, pp. 1097–1105, 2012.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015.

[24] Google, "Google trends: Pytorch and tensorflow." [Online]. Available: https://trends.google.co.uk/trends/explore?date=today%205-y&q=%2Fg%2F11gd3905v1,%2Fg%2F11bwp1s2k3

[25] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[26] D. Zhang, X. Dai, and Y.-F. Wang, "Dynamic temporal pyramid network: A closer look at multi-scale modeling for activity detection," *arXiv preprint arXiv:1808.02536*, 2019.

[27] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," Dec. 2014.

[28] 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*. Springer, 2016, pp. 20–36.

[29] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017. [Online]. Available: https://arxiv.org/abs/1703.05175