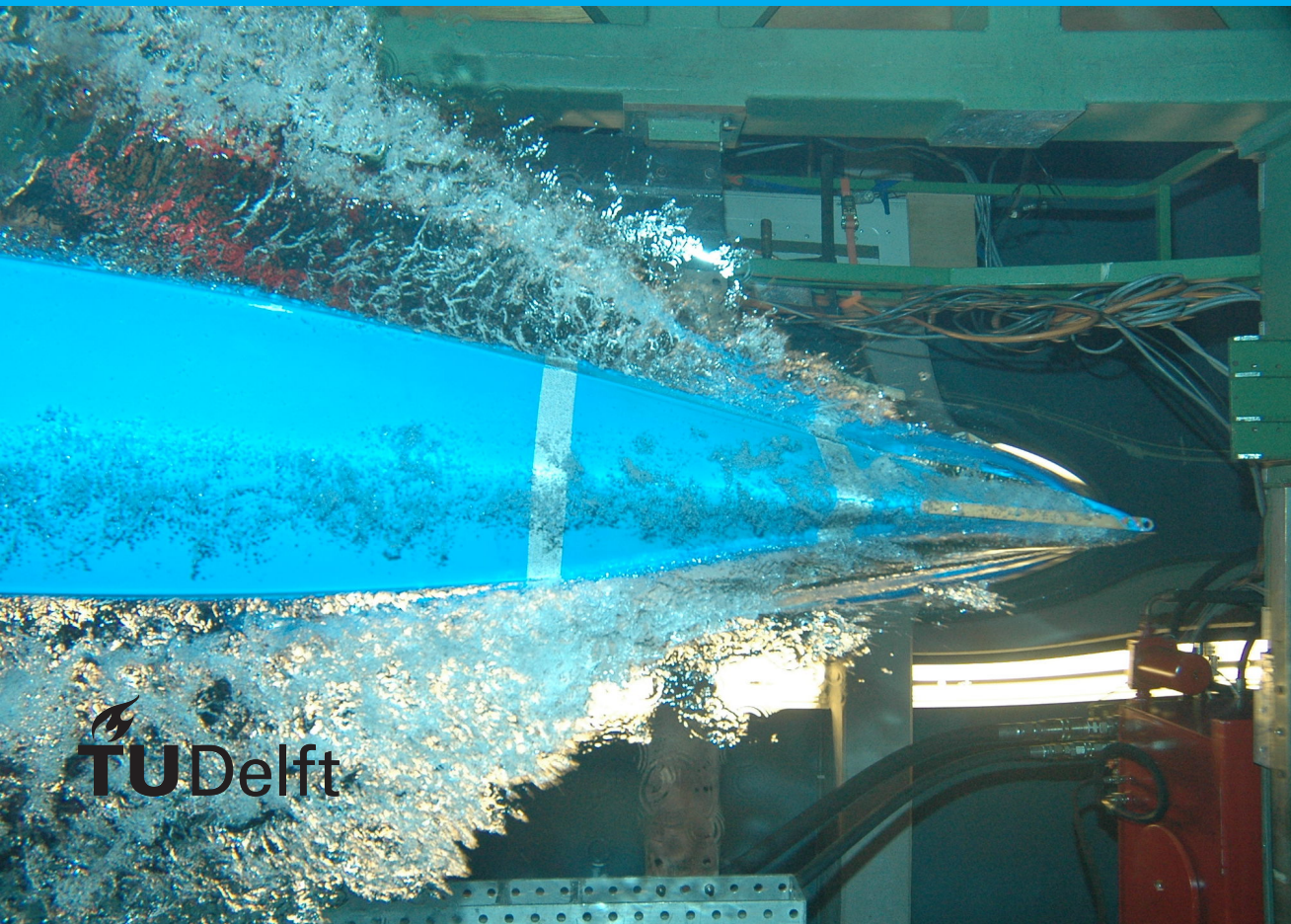
Assessment of Parkinson's Disease Severity from Videos using Deep Architectures

Zhao Yin



TU Delft

ASSESSMENT OF PARKINSON'S DISEASE SEVERITY FROM VIDEOS USING DEEP ARCHITECTURES

by

Zhao YIN

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 19, 2020 at 2:00 PM.

Student number:	4778863	
Project duration:	October, 2019 - August, 2020	
Thesis committee:	Dr. J.C. van Gemert,	TU Delft, supervisor
	Dr. Hamdi Dibeklioglu,	Bilkent University, daily supervisor
	Dr. Huijuan Wang,	TU Delft, external committee member
	Victor Geraedts,	LUMC, external expert
	Ziqi Wang,	TU Delft, co-supervisor

An electronic version of this thesis is available at http://repository.tudelft.nl/.

PREFACE

The thesis report presents the work done for my master's thesis project. The research was conducted within the Computer Vision Lab, TU Delft, under the supervision of Dr. J.C. van Gemert. Dr. Hamdi Dibeklioğlu and Ziqi Wang have been my daily co-supervisors.

In the first part, the report shows the publication output of this work - the scientific paper, which contains the main contents of the report, including motivation, related work, methods, experiments, and results. The latter part describes some fundamental knowledge related to this work to help understand the scientific paper.

During the project, I have gained a lot, not only the knowledge for the research but also the time management skills that help me complete the project systematically. I believe those skills will still affect me greatly in my future career.

I want to thank Dr. J.C. van Gemert for his great support and useful guidance throughout the project. Furthermore, I would like to acknowledge Dr. Hamdi Dibeklioğlu, Ziqi Wang, and Victor Geraedts for daily help on my thesis's progress and always being patient with my mistakes. Dr. Hamdi Dibeklioğlu and Ziqi Wang helped me construct the structure of the technical parts of the project. Victor Geraedts helped me understand difficult clinical contents. I would also like to thank my family and friends for their encouragement and support.

Zhao Yin
Delft, August 2020

CONTENTS

1

SCIENTIFIC PAPER

Assessment of Parkinson's Disease Severity from Videos using Deep Architectures

Zhao Yin
Delft University of Technology
zhao_yin@outlook.com

***Abstract*— Parkinson's disease (*PD*) diagnosis is based on clinical criteria, i.e. bradykinesia, rest tremor, rigidity, etc. Assessment of the severity of PD symptoms, however, is subject to inter-rater variability. In this paper, we propose a deep learning based automatic PD diagnosis method using videos recorded during the assessment with the Movement Disorders Society - Unified PD rating scale (*MDS-UPDRS*) part III. Seven tasks from the MDS-UPDRS III are investigated, which show the symptoms of bradykinesia and postural tremors. We demonstrate the effectiveness of automatic classification of PD severity using 3D Convolutional Neural Network (*CNN*) and the PD severity classification can benefit from non-medical datasets for transfer learning. We further design a temporal self-attention (*TSA*) model to focus on the subtle temporal vision changes in our PD video dataset. The temporal relative self-attention-based 3D CNN classifier gives promising classification results on task-level videos. We also propose a task-assembling method to predict the patient-level severity through stacking classifiers. We show the effectiveness of TSA and task-assembling method on our PD video dataset empirically.**

***Index Terms*— Parkinson's disease (PD), deep learning, transfer learning, self-attention, multi-domain learning.**

I. Introduction

Parkinson's disease (PD) is a chronic, progressive neurological disorder, affecting over 10 million people around the world according to the American Parkinson Disease Association (APDA) [42]. Individuals with Parkinson's disease typically present with characteristic motor symptoms, including bradykinesia (i.e. slowness of movement), rigidity (stiffness), and rest tremor [45]. These symptoms are progressive over time, subsequently leading to an increase in their severity.

At present, the Movement Disorder Society - Unified Parkinson's Disease Rating Scale (MDS-UPDRS), containing four parts: I for non-motor experiences of daily living, II for motor experiences of daily living, III for motor examination and IV for motor complications, has been widely used as a validated tool to quantify PD severity [18], [31]. MDS-UPDRS is the revised and more comprehensive version of the original UPDRS [16] and they are highly correlated on the motor sections [32]. This study uses the MDS-UPDRS part III (MDS-UPDRS-III) as the measurement for analysis, which contains 18 tasks and 33 scores, with some tasks pertaining to either left or right extremities. Each task, tied to a symptom assessed by clinically trained raters, has five responses linked to symptom-severity: 0-normal, 1-slight, 2-mild, 3-moderate, and 4-severe, providing consistency across tasks. Collapsing all the scores to provide the patient with a composite total score is not recommended by [18] but can still be applicable given the minimal clinically important difference threshold values [30] and is often used in clinical practice to monitor disease progression. Although MDS-UPDRS-III is currently the gold standard to quantify the severity, it still has the potential to cause less reliable ratings due to the intrinsic inter-rater variability caused by the non-identical inter-rater protocols and inexperienced examiners [15], [48]. Besides, the presence of the specialist is mandatory when giving the rating decisions. However, the patient's motor status can change significantly during the two consecutive examinations. These difficulties make the manual rating inefficient and urge for automatic quantification method. In this work, we propose a deep learning based PD severity quantification approach using videos. We show the overall pipeline in Fig. 1.

The goal of PD severity quantification is that, given an individual patient's video performing a specific task, the corresponding severity level can be predicted by the machine learning algorithm to assist ratings of examiners. As the task performed by the patient in the video is a kind of action, we naturally think of the human action recognition method to solve the identification of Parkinson's severity. Recently, many action recognition architectures [5], [17], [19] achieve promising performance on public human action datasets and one of the mostly used architecture is the inflated 3D CNN (I3D) [5], which is a 3D CNN with 3D kernels inflated from a 2D CNN with an additional temporal dimension. Therefore, we opt to use I3D as the base model for this work.

Due to the small size of our PD dataset, directly training I3D from scratch is inefficient and prone to overfitting; thus, we use transfer learning to pre-train the network on large datasets to make the training process more stable. However, public datasets we pre-train on have noticeable motion differences while the motion difference in our PD dataset is subtle. Such large domain discrepancy makes it difficult to transfer knowledge between domains, so we need a solution to focus on exploring the temporal motion changes. Besides, the video in our dataset is a repeating task with periodic actions, where the model should learn the repeating frequency or the starting and ending point. Thus, we need another solution to assign different weights for the frames of the video. Additionally, as stated in [37], [39], not all frames are equally crucial for action recognition, so we propose to use temporal self-attention to

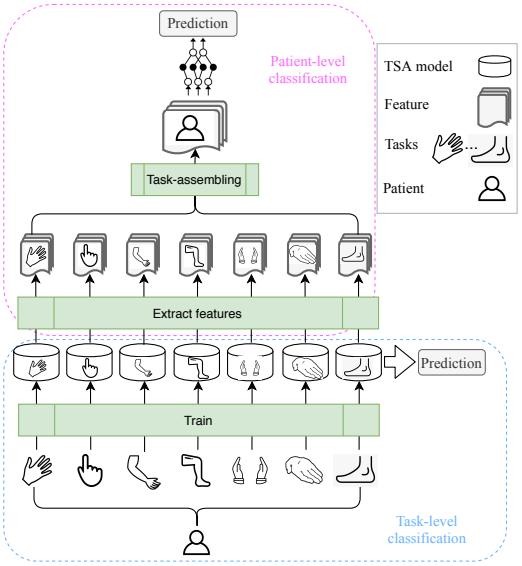


Fig. 1: The flowchart of the automatic PD severity quantification. The task symbols from left to right denote task *finger tapping*, *hand movements*, *kinetic tremor*, *leg agility*, *postural tremor*, *pronation*, and *toe tapping*.

assign the weights for frames as well as solve the domain discrepancy issue. The benefit is not only for such a repeating dataset but also for other datasets because it holds for other datasets as well that not all frames are equally important.

Once we can predict each task's severity, each patient will have a separate severity score for each task. However, it's more clinically interesting to give a summary severity for the patient rather than multiple ones, so we propose to apply a novel task-assembling method to combine the predictions of different tasks from the patient to predict a single score.

The contributions of this work are:

1) we perform automatic task-level PD severity classification using I3D from videos of our PD dataset, based on seven tasks in MDS-UPDRS-III;
2) we show that I3D can benefit from non-medical datasets with transfer learning;
3) we propose TSA to overcome the large discrepancy of motion difference between non-medical datasets and our PD dataset during transfer learning;
4) we propose a task-assembling method to combine the models of different tasks to produce a single concluding severity score for a patient.

II. Related Work

A. Machine/Deep Learning Based Approaches

Machine/deep learning based PD analysis has been intensively researched in recent years. For instance, the K-nearest neighbors (KNN) ensemble AdaBoost classifier and support vector machines (SVM) with RBF kernel are used to classify between PD patients and controls based on the features extracted from individual handwriting [14]. For image-based analysis, discriminant analysis and SVM are used to differentiate PD patients and controls based on segmented high-activity regions of the single photon emission computed tomography (SPECT) scans of the brain [4]. For signal-based analysis, signals acquired from the gyroscope attached to the subject's finger are extracted to feed into multiple classifiers [46]. In [2], glottal flow features are used as input for SVM classifier to detect PD with an accuracy of 75.3%. Apart from the image- and signal-based analysis, the video is also used as an input data type for PD quantification [51], [54]. However, to the best of our knowledge, apart from [43] in which freezing of gait videos are used to feed the 3D network, most researchers extract the feature from videos as the final input for classifiers without fully utilizing the video resource. Based on machine/deep learning approaches, our work applies action recognition method to quantify PD severity using RGB video data.

B. Transfer Learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [53]. It is widely used as a pre-training approach to offer the model a better starting point instead of training from scratch. In the work of [34], CNN layers trained from ImageNet is reused to transfer visual recognition tasks to learn mid-level representations for small datasets. In action recognition, researchers apply transfer learning to pre-train the model on a large dataset to make the training process faster, more efficient, and less prone to overfitting with a significant performance improvement [5], [19]. Most related research shows that transfer learning can be a useful tool to make the network work on small datasets, and thus we use transfer learning in this work to help improve the performance on our PD dataset.

C. Capture Temporal Information

1) For General Video Dataset: In action recognition, researchers apply various methods to capture the temporal information crucial in video data. In the work of [47] (*C3D*), 3D CNN is used as a spatiotemporal feature extractor for videos, and the extracted features are used as inputs for simple linear classifiers. Based on the 3D CNN, an I3D is introduced to take advantage of pre-trained 2D models [5]. Similar to 3D CNN, I3D performs 3D convolution on both spatial and temporal dimensions simultaneously. However, in I3D, pre-trained 2D filters are repeated or inflated multiple times to form 3D filters. Therefore, I3D can benefit from successful image (2D) classification models trained on large datasets such as ImageNet [11]. Besides 3D CNN, a combination of a stack of CNNs and Long-Short Term Memory (LSTM [21]) networks is applied to exploit the temporal information [1], [12] as well. These methods apply either 3D CNN or 2D CNN with fusion methods such as LSTM on the video data to capture the temporal information. We use I3D as our base model because of its decent performance on public datasets, including Kinetics-400 experimented in [19].

2) For Periodic- and Subtle-Motion Video Dataset: The spatiotemporal template of motion features is used to recognize and segment the repetitive motion by template matching [36]. In [9], CNN is used to count the number of repetitions, and circle length in periodic-motion videos. Besides the task of action recognition, the estimation of repeating frequency is studied in [35], using a Lagrangian approach and an Eulerian approach as the frequency estimators. In periodic-motion videos, we need to focus on the repeating frequency, starting, and ending points to make the model work.

In medical datasets such as movement disorder dataset, videos usually have subtle motion changes, which are hard for architectures to work because subtle motion information is difficult to capture and can not even be seen with bare eyes. The subtle motions can be magnified using a steerable pyramid [26], [50]. In the work of [10], motion frequency is used to estimate material properties. Similarly, signal analysis in the Fourier domain is employed to estimate the tremor frequency of subtle motions [35]. In subtle-motion videos, we need to focus on magnifying the subtle motion or directly estimating the frequency.

D. Self-Attention

Self-attention is extensively explored since the Transformer network is introduced for machine translation [49] where the self-attention is used to compute the interactions between words. In recent work, the QANet [55] architecture uses self-attention in cooperation with convolutions for machine-reading and question answering tasks, where the convolution computes local interactions and self-attention computes global interactions. In image tasks, self-attention with relative positional embeddings is usually used to compute the interactions among pixels in the same image and allows the model to learn which part of the image is of more importance [3]. In the non-local network [52], self-attention can be used in convolutional architectures to learn the long-range interactions among pixels in images or videos for object detection and video classification. In general, self-attention is used in architectures for modeling sequences as it can capture long-distance interactions. In this paper, we propose a new method, temporal self-attention model, for PD quantification, which involves I3D and the self-attention mechanism, attempting to detect the periodic and subtle motion in the video data.

E. Multi-domain Learning

Multiple similar domains can be learned to let the model work on a new target domain using parameter combination from multiple classifiers [13], [24]. In [6], perceptron-based algorithms are employed for multi-task binary classification problem with the similarity estimation among tasks. Basically, multi-domain learning explores the relationship between tasks or domains and incorporates them to solve a new task. In this work, we combine the features from multiple domains (i.e., tasks from MDS-UPDRS-III) to predict patient-level PD severity classification.

III. Methods

In this section, we explain the details of the proposed video-based automatic classification of Parkinson's disease. The overall flow of the algorithm is described as follows. Initially, each video is preprocessed to be consistent with other videos concerning size, task starting point, mean, and standard deviation. At the same time, we use network-based transfer learning to transfer knowledge from non-medical datasets to the medical one, i.e., reusing the network trained on large datasets as the pre-trained model to replace model initialization. Then, the pre-trained model is fine-tuned on the collected Parkinson's dataset to learn the underlying patterns. After fine-tuning, the model can be used as the classifier for task-level classification. By combining the features extracted by the deep models from different tasks and training a shallow neural network using those features, patient-level analysis can be further made.

A. Inflated 3D Convolutional Neural Network (I3D)

In this paper, we use I3D as the base network with ResNet as the backbone instead of Inception-v1 as the former is more natural to extend (currently 18, 34, 50, 101, 152-layer variations are available) and its pre-trained models are already available [19]. Furthermore, rather than using two streams (RGB frames and optical flow), we use RGB frames as the only input because computing optical flow is time-consuming, which is not feasible if the real-time prediction is required.

The model is optimized using gradient descent by minimizing the empirical loss with class-balanced focal loss [8]:

$$J(\omega) = \frac{1}{N}\sum_{i=1}^{N}\left(-\frac{1-\beta}{1-\beta^{n_y}}\sum_{c=1}^{C}(1-p_{i,c}^{t})^{\gamma}\log(p_{i,c}^{t})\right)+\lambda\left\|\omega\right\|_2^2, \tag{1}$$

where C, N, ω and γ denote the number of classes, number of samples, learned parameters and *focusing* parameter, and $\beta = (N-1)/N$. n_y stands for the number of samples in the ground-truth class y and p^t is defined as

$$p^t = \begin{cases} p & \text{if } y = c \\ 1-p & \text{otherwise.} \end{cases} \tag{2}$$

B. Self-attention Replacing Convolution

We describe the proposed temporal self-attention block for video classification following the symbol styles of [3], that is, T, H, W and C refer to the time, height, width and number of input filters of an activation map. N_h, d_v and d_k refer to the number of heads, the depth of values and the depth of keys and queries in multi-head-attention [3], assuming that d_v and d_k can be evenly divided by N_h. d_v^h and d_k^h denote the depth of values and keys/queries per attention head.

1) Temporal Self-attention over Video Volume: Given an input tensor of shape (C, T, H, W) with the batch dimension omitted for simplicity, we first transpose and flatten it to a tensor $X \in \mathbb{R}^{HW\times T\times C}$ and perform multi-head-attention

on the temporal dimension. The output of the temporal self-attention block for a single head is

$$O_h = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k^h}}\right)V, \tag{3}$$

where queries $Q = XW_q$, keys $K = XW_k$ and values $V = XW_v$ and W_q, $W_k \in \mathbb{R}^{C\times d_k^h}$ and $W_v \in \mathbb{R}^{C\times d_v^h}$ are learned linear transformations[1]. Note that we transpose the last two dimensions of V to correctly multiply with Q. Concatenating the outputs from all heads we get

$$O = \left[O_1, \ldots, O_{N_h}\right]. \tag{4}$$

The shape of O is $(HW \times T \times d_k^h)$. We then project it again using the learned linear transformation $W^O \in \mathbb{R}^{d_v\times d_v}$

$$\text{MultiHead}(Q, K, V) = OW^O, \tag{5}$$

where $\text{MultiHead}(Q, K, V)$ is of shape $(HW \times T \times d_v^h)$. After reshaping back to the original spatial and temporal dimension, we have the final output $\text{MultiHead}(Q, K, V) \in \mathbb{R}^{T\times H\times W\times d_v}$ of our temporal self-attention block if relative postional embeddings [3] (see Section III-B.2) not applied.

The novelty of our temporal self-attention block is applying the self-attention mechanism solely on the temporal dimension, leaving the spatial dimension untouched. The advantage is that self-attention can capture the long-range temporal changes while keeping standard CNN there, capturing the necessary visual patterns simultaneously. As such, the abilities of both self-attention and CNN can remain and incorporate in the temporal self-attention block, which effectively makes up the drawback of I3D.

Fig. 2 illustrates the temporal self-attention mechanism. The temporal sequence of feature points (red ones) that share the same spatial position is the atomic unit, on top of which the temporal self-attention applies. We have HW sequences/units located at all spatial positions, and each of them is independent of others when performing the temporal self-attention.

2) Relative Positional Embeddings: The only difference between 1D and 2D relative positional embeddings is the dimensions involved in the algorithm. Thus we refer to [3] for the details of 2D relative positional embeddings, and we do not discuss the 1D variation anymore in this paper. To implement temporal relative self-attention, we add relative temporal information to the temporal self-attention block's output. The output is now changed from Equation 3 to

$$O_h = \text{Softmax}\left(\frac{QK^T + S_T^{rel}}{\sqrt{d_k^h}}\right)V, \tag{6}$$

where $S_T^{rel} \in \mathbb{R}^{HW\times T\times T}$ is the matrix of relative position logits along the temporal dimension.

3) Temporal Relative Self-attention: We combine temporal self-attention with 1D relative positional embeddings to form our new building block-temporal relative self-attention block. Fig. 3 describes the whole pipeline of the proposed block.

[1]Bias terms are ignored when we mention linear transformations.

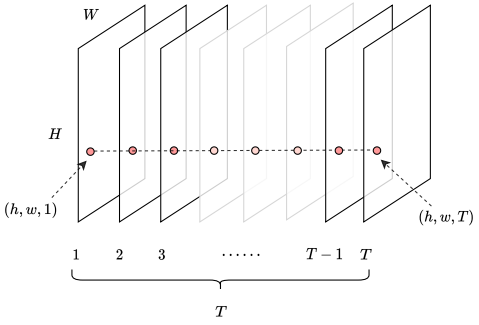


Fig. 2: An example of temporal self-attention. Assume the stack of those rectangles is a feature map (or more intuitively for 3D data, feature volume) from one channel. Each rectangle represents the spatial visual patterns at a specific temporal position. Our temporal self-attention is performed on the feature points colored in red, which share the same spatial position along the temporal dimension. It can be seen as self-attention through time.

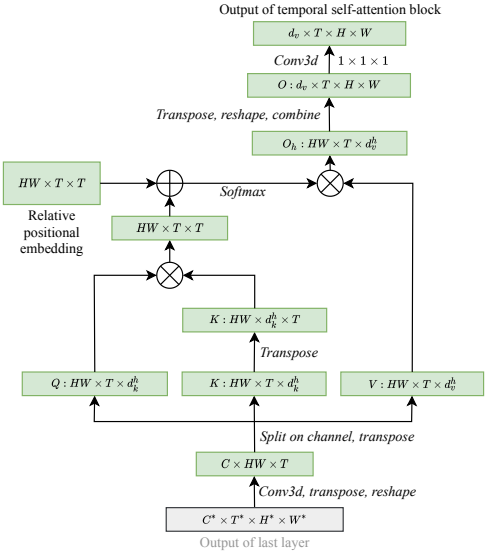


Fig. 3: The general pipeline of our temporal relative self-attention. Rectangles in the workflow represent tensors with shape specified, and italic words stand for tensor operations. $\otimes$ and $+$ denote tensor product and addition.

4) Temporal Relative Self-attention Network (TSA): Once the temporal relative self-attention block is built up, the convolutional block in any architecture can be substituted. Take 3D ResNet-34 for instance, which has 33 convolutional layers. We replace those layers as many as possible with our block from the last convolutional layer to the first one until we hit the memory bottleneck.

The time complexity of our block is $O(HWT^2d_k)$ compared to the convolutional block $O(HWTC)$, which is time-efficient since the temporal size is typically small after a few layers. The memory cost is $O(HWT^2N_hd_k^h)$ compared to the convolutional block $O(HWTC)$.

C. Multi-task Assembling

Using the model we discussed in previous sections, it can solve the task-level severity classification on our PD dataset. Given a sample related to a specific task from the dataset, we can predict its task severity S_t. Nonetheless, it is more clinically interesting to tell the severity score of a patient S_p instead of tasks. Therefore, we propose two multi-task assembling methods to combine the tasks to do severity classification for patients. Note that the following methods require trained models on the PD dataset for task-level classification.

1) Vector Averaging and Vector Weighting: We use the trained model as a feature extractor to compress the information of a video into a dense one. We first extract the flattened vector $F \in \mathbb{R}^d$ of dimension size d as the compressed information, which is the input feature of the fully connected layer. Each video, containing only a single task from a patient, produces one feature vector F_m of task m and all videos from that patient produce feature vectors $F_M \in \mathbb{R}^{d\times M}$ of all M tasks. Different tasks may contribute unequally to a patient's severity score, so we use two strategies to convert (or combine) F_M into a vector $F \in \mathbb{R}^d$, representing the feature of a patient.

The first approach is to average features, formulated as

$$F = \frac{1}{M}\sum_{m=1}^{M} F_m, \tag{7}$$

by assuming each feature (task) contributes equally. The second approach is to take the weighted average of features as the following

$$F = \frac{1}{M}\sum_{m=1}^{M} \alpha_m F_m, \tag{8}$$

where α_m ($\sum_{m=1}^{M}\alpha_m = 1$) is the learnable weight for task m. The first approach is a special case of this one. Afterward, F is fed as input to train a shallow neural network[2]. The network is optimized using gradient descent by minimizing the empirical loss $J(\omega)$ (see Equation 1) where N is the number of patients.

2) Attention-based Feature Weighting: In the feature averaging and weighting approach, we assume task weights are identical across all patients. However, patients may not share the same task weights so that the global task weights may be insufficient and inaccurate. Therefore, we propose to use channel-wise attention-based weighting, which automatically assigns task weights for each patient separately. To do so, we use another feature map $F_M \in \mathbb{R}^{M\times C\times T\times H\times W}$ (M denotes the number of tasks), the output of the last convolutional or our self-attention layer, as the extracted feature for a video.

The first weighting strategy is to apply squeeze-and-excitation block [22] to map the input feature F_M to a set of channel weights. As the task weights are our concerns instead of the channels, we take the task dimension as the channel dimension in the squeeze-and-excitation block. The process can be formulated as follows. First, squeeze global information into a task descriptor by using global average pooling to generate task-wise statistics

$$z_m = \frac{1}{C\times T\times H\times W}\sum_{c=1}^{C}\sum_{t=1}^{T}\sum_{h=1}^{H}\sum_{w=1}^{W} F_m(c,t,h,w), \tag{9}$$

where F_m denotes the feature map for task m. Then we excite the task-wise statistics to task weights ($W_1 \in \mathbb{R}^{\frac{M}{r}\times R}$, $W_2 \in \mathbb{R}^{R\times\frac{M}{r}}$ in which r is the dimensionality-reduction ratio)

$$\alpha_M = \sigma(W_2\delta(W_1 z_M)), \tag{10}$$

where α_M, σ and δ denote task weights, the sigmoid activation and the ReLU [33] function. Finally we obtain the combined feature map $F \in \mathbb{R}^{C\times T\times H\times W}$

$$F = \frac{1}{M}\alpha_M F_M. \tag{11}$$

Applying the squeeze-and-excitation block to get task weights is rather simple but turns out to be efficient. It flexibly generates different weights for different patients accordingly. However, this approach assumes each feature point in the feature map contributes equally, which means a task weight is a global weight for all feature points. We can explore even further by making each feature point having its own weight $\alpha_{t,h,w,m}$, which brings about the pixel-wise attention-based weighting approach.

We opt to use the self-attention mechanism similar to our temporal relative self-attention block for pixel-wise weighting, by applying it on the task dimension instead of the temporal dimension. First, we reshape and flatten $F_M \in \mathbb{R}^{M\times C\times T\times H\times W}$ into the shape of $(THW\times M\times C)$ and then the output of a single attention head can be computed as

$$O_h = \text{Softmax}\left(\frac{(F_M W_q)(F_M W_k)^T}{\sqrt{d_k^h}}\right)(F_M W_v), \tag{12}$$

where W_q, $W_k \in \mathbb{R}^{C\times d_k^h}$ and $W_v \in \mathbb{R}^{C\times d_v^h}$ are learned linear transformations. Afterwards, we combine attention results of all heads and project using $O^W \in \mathbb{R}^{d_v\times d_v}$ to form the task weighted feature map

$$F = [O_1,\ldots,O_{N_h}]O^W. \tag{13}$$

Note that the task weights for each feature point $\alpha_{t,h,w,m}$ is implicitly embedded in the computation of attention output.

Task weighted features using both approaches are fed into a shallow neural network consisting of batch normalization [23], the ReLU function, global average pooling, and a fully connected layer.

[2]0, 1 or 2 hidden layers with non-linear activation.

The summary of the proposed four task-assembling methods can be found in TABLE I. Vector averaging and vector weighting use the outputs of the last global average pooling layer while attention-based weighting methods use the outputs of the last convolutional/self-attention layer in the network. We denote *avgpool* and *layer4* as the feature types.

IV. Experimental Settings

A. Dataset

In this paper, we introduce a new video dataset for Parkinson's disease analysis. We develop this dataset principally because there is a lack of such datasets for Parkinson's disease analysis. We believe that having one will facilitate research in this area because the dataset simulates the procedure of how experts diagnose patients using MDS-UPDRS-III scores. Besides, the dataset is challenging enough to act as a performance benchmark where the advantages of different architectures can be demonstrated.

1) Data Collection: Consecutive patients who underwent either a Levodopa Challenge Test (LCT [38], [40]) prior to DBS surgery, or underwent a Stimulator Challenge Test (SCT, [7], [20]) after DBS surgery, were recruited. All patients fulfilled the criteria for idiopathic PD. Patients who underwent a LCT were videotaped twice (i.e. Med-OFF and Med-ON); patients who underwent SCT were videotaped three times (Med-OFF-Stim-ON [28], Med-OFF-Stim-OFF [29], Med-ON-Stim-ON [20]). Video recordings were made with the camera in a fixed position, with a complete overview of the patient central on the screen. Due to the varying nature of the examination room, the camera's position and angle towards the patient varied, as well as the background and surroundings. During the MDS-UPDRS-III examination, the zoom-function was occasionally used to focus on the hands or feet.

All videos were made in one continuous recording of the examination. Separate segments were created by clipping the videos per task (left and right separately if required): bradykinesia of the hands (MDS-UPDRS-III items 3.4, 3.5, 3.6), bradykinesia of the legs (items 3.7, 3.8), postural tremor (item 3.15), kinetic tremor (item 3.16). Rigidity was not included as this symptom is not assessed through visual observation; global bradykinesia, speech, freezing-of-gait, and rest-tremor were not included as no specific video-segment pertained to those tasks and they were evaluated throughout the entire recording. The local medical ethics committee waived the formal evaluation of the study. All patients gave written informed consent.

2) Dataset Overview: The dataset contains 39 subjects and 1082 videos. Each sample in the dataset is of resolution 1920 by 1080 and 25 fps. The duration of samples may be different on different tasks. Fig. 4a shows the duration distribution of our dataset.

The dataset contains $T = 11$ tasks for most of the patients based on the MDS-UPDRS-III, namely finger tapping, gait freezing, hand movements, leg agility, pronation, toe tapping, arising from chair, kinetic tremor, postural tremor, postural stability and rest tremor. Note that not all tasks are used in the experiments. Each video has a task-level severity score $S_t \in \{0, 1, 2, 3, 4\}$ (0: normal, 1: slight, 2: mild, 3: moderate and 4: severe) labeled by experts. Each patient has a patient-level severity score

$$S_p = \sum_{t=1}^{T} S_t. \tag{14}$$

The distributions of S_t (over all tasks) and S_p are shown in Fig. 4b and Fig. 4c.

B. Settings

To evaluate our methods for Parkinson's severity classification, we use the above-described dataset. In our experiments, only RGB frames are used as the input for the deep architectures. The clips are resized to $32 \times 224 \times 224$ resolution without changing their spatial aspect ratios.

After splitting the dataset into five folds, we train networks on four of them and test it on the rest one in rotation. The overall accuracy is obtained by taking the average of the individual accuracy tested on each fold. There is no subject overlap between folds to avoid network cheating by recognizing the appearance of the patient.

I3D is pre-trained on both UCF-101 (by ourselves) and Kinetics-400 (by [19]). TSA is pre-trained only from UCF-101 (by ourselves). Batch size of 15, learning rate of 0.001 without decay and weight decay (λ) of 0.01 are used.

The task-level score $S_t \in 0, 1, 2, 3, 4$ is split into two classes: class 0 for $\{0, 1\}$ and class 1 for $\{2, 3, 4\}$ since we are more interested in whether the model can distinguish between the slight and severe group of patients. The patient-level score S_p is split into three classes in the way that each class has an equal number of patients. Method specific settings are provided alongside when showing the results in Section V.

V. Results

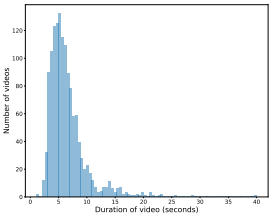
In this section, we show the results of our experiments. We test seven tasks with high quality videos, *finger tapping*, *hand movements*, *kinetic tremor*, *leg agility*, *postural tremor*, *pronation* and *toe tapping*. They are denoted as *finger*, *hand*, *kinetic*, *leg*, *postural*, *pronation* and *toe* for simplicity. We use ResNet-34 as backbone because through experiments we find that ResNet-34 is the most suitable one in this study, considering the size and difficulty of our dataset. One can of course use other backbones if the size, complexity and classes of the dataset are different from ours. We have to emphasize that, in all experiments, although patients contribute more than one video, no patient is included into both the training- and test-set because even though videos of a patient are separate ones, they are still from the same patient.

A. Validate Temporal Relative Self-attention Network

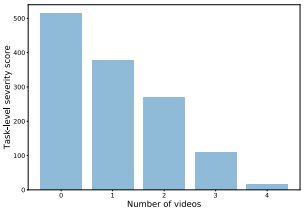
Before applying TSA on PD dataset, we first check whether it works better than I3D on two frequently used public datasets UCF-101 and HMDB-51. Hyper-parameters are chosen without optimization: input shape of $64 \times 224 \times 224$, *lr* of 0.001, batch size of 45, weight decay of 10^{-5} and optimizer of SGD with momentum [44]. The backbone is ResNet-18 for fast

TABLE I: The summary of four task-assembling methods.

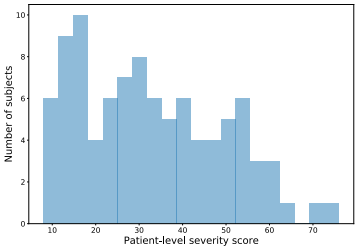
	vector averaging	vector weighting	channel-wise attention weighting	pixel-wise attention weighting
Input type	*avgpool*	*avgpool*	*layer4*	*layer4*
Weights differ among tasks	✗	✓	✓	✓
Weights differ among patients	✗	✗	✓	✓
Weights differ among feature points	✗	✗	✗	✓
Core mechanism	averaging	learnable weight vector	squeeze-and-excitaion [22]	self-attention



(a) The histogram of the duration of samples, using 80 bins. The average duration is 6.3 seconds, and 90% of samples are shorter than 10 seconds, with less than five samples longer than 25 seconds.



(b) The bar chart shows the distribution of task-level severity score. From low to high severity class, the number of samples decreases, which shows the class imbalance issue in our dataset.



(c) The histogram of patient-level severity score using 20 bins. Compared to task-level severity distribution, patient-level severity distribution has no obvious imbalance issue. The number of patients across the range of severity is approximately on the same level.

Fig. 4: Distributions of the sample duration and task/patient-level severity of our dataset.

TABLE II: Top-1 accuracy on UCF-101 and HMDB-51. All accuracy are averaged over three splits. Both methods use ResNet-18 as the backbone. TSA shows better performance on both datasets so that it can be further applied to PD dataset.

Method (scratch)	UCF-101	HMDB-51
ResNet-18 [19]	42.4	17.1
TSA ResNet-18	51.5	22.1

illustration. TABLE II shows that TSA outperforms I3D when both trained from scratch. The performance improvements demonstrate the effectiveness of TSA and the possibility of applying it to our PD dataset.

B. Benefit from Transfer Learning

We utilize three datasets: Kinetics-400 [25], UCF-101 [41] and HMDB-51 [27] to pre-train our models considering their large sizes, high quality and popularity. Then, we fine-tune the pre-trained models on our PD dataset. Since our dataset contains periodic and subtle motions while public datasets have easily distinguishable motions, the relatedness between our dataset and public datasets is not tight. As such, the parameters from the convolutional stem may not be optimal after transferring to our dataset. Thus all layers of the model rather than part of them are fine-tuned.

I3D and task *finger* and *hand* are used to demonstrate the function of transfer learning. Convergence is confirmed for every compared setting for a fair comparison. Note that for task-level classification we have binary classes. In TABLE III, I3D trained from scratch, I3D pre-trained from UCF-101, and I3D pre-trained from Kinetics-400 are compared based on the binary accuracy, precision, recall, and mean f1-score. Here the mean f1-score is formed as:

$$\frac{2}{C}\sum_{i=1}^{C}\frac{\text{precision}_i \times \text{recall}_i}{\text{precision}_i + \text{recall}_i}, \tag{15}$$

where C is the number of classes. In general, I3D pre-trained from the two datasets outperform I3D (scratch), demonstrating that I3D can benefit from non-medical datasets with transfer learning. Moreover, the performance improvement of I3D (Kinetics-400) from I3D (scratch) is more notable than I3D (UCF-101) especially on task *hand*, which indicates the model would benefit more from a larger dataset with transfer learning.

C. Task-level Severity Classification

Building a model good at predicting the task severity score is our first concern and affects the later experiments and research. Two architectures - I3D and our TSA are compared in TABLE V on seven tasks from MDS-UPDRS-III. The class distribution can be found on TABLE IV. Note that we replace convolutional layers in 3D ResNet-34 *layer3* and *layer4* with temporal relative self-attention block to construct our TSA network. The dataset in the brackets denotes on which the model is pre-trained. We show precision and recall along with mean f1-score because the mean f1-score as the only indicator can be insufficient in some cases. For instance, when either precision or recall is too low, and the other is sufficiently high, we can not conclude that this result is satisfactory by only focusing on the moderate mean f1-score averaged on the two extreme values. In general, the mean f1-score is used as the

TABLE III: Accuracy, precisons, recall and f1-score on two tasks from MDS-UPDRS-III using I3D with and without transfer learning. I3D using transfer learning achieves better results than I3D trained from scratch on both *finger* and *hand* tasks. Moreover, transfer learning with a larger dataset (i.e., Kinetics-400) has more benefits to the model.

Method	Metric	*finger*	*hand*
I3D (scratch)	acc	65.4	65.6
	precision, recall	0.60, 0.70	0.69, 0.61
	f1	0.65	0.65
I3D (UCF-101)	acc	68.6	70.0
	precision, recall	0.55, 0.76	0.76, 0.61
	f1	0.66	0.68
I3D (Kinetics-400)	acc	69.2	77.5
	precision, recall	0.57, 0.76	0.80, 0.75
	f1	0.67	0.77

TABLE IV: Class distribution of seven tasks in our Parkinson's dataset. In general, the class imbalance in task *finger*, *hand*, *pronation* and *toe* is acceptable. In remaining tasks, the class imbalance issue is severe.

Task	Class 0	Class 1
finger	66	91
hand	89	71
kinetic	130	38
leg	145	39
postural	62	23
pronation	104	72
toe	87	71

indicator when both precision and recall are reasonably good, otherwise considering inspecting precision and recall.

1) Task-level Performance: TABLE VI extracts the best mean f1-score of all seven tasks from TABLE V regardless of the methods used and Fig. 5 shows the receiver operating characteristic (ROC) curve for each task accordingly. Three out of seven tasks have a mean f1-score higher than 0.7 and close to 0.8, and only one task *leg* is under 0.6. The average mean f1-score across all seven tasks is 0.71, sufficiently good for classification on a medical dataset. It demonstrates that deep architectures can predict the task (i.e., task from MDS-UPDRS) severity of a patient with decent accuracy given the video from that task.

In particular, task *finger*, *hand* and *pronation* are the top-3 well-classified task in terms of mean f1-score and ROC curves in Fig. 5a, 5b and 5f, because 1) most of the videos are zoomed in to focus on the objects, making it easier for the model to look at the relevant patterns and 2) the class imbalance problem is slight compared to task *kinetic*, *leg* and *postural*. The difficulty and label noise of tasks can also contribute to the phenomenon but no argument can be made without confirmation from experts. On the opposite, task *leg* has the lowest mean f1-score, and the ROC curve in Fig. 5d does not bulge towards the top-left corner of the figure, indicating a corrupt model for task *leg*. After inspecting TABLE V, we can observe quite low recalls of 0.17 and 0.14 using I3Ds and an inadequate recall of 0.35 using TSA.

The performance discrepancy between tasks exposes some disadvantages of our architectures. First, the ratio of objects, e.g., hand in task *hand movements* and toe in task *toe tapping*, occupying the bounding box of the video matters. In task *finger tapping*, *hand movements* and *pronation*, the zoom-function is occasionally used to focus on the objects, and most of the videos are zoomed in during the pre-processing stage, which gives the architectures cleaner and more easy-to-identify input data. Second, the effects of the class imbalance problem on the architectures cannot be ignored. Due to the PD dataset is a periodic- and subtle-motion dataset, which is different from public datasets. Identifying task severity is harder than classifying different human actions. In such a case, the extreme class imbalance can corrupt the architectures' behavior even if the class-balanced loss [8] is adopted. However, the class imbalance is everywhere in real-world settings or at least in Parkinson's disease. As such, we leave solving class imbalance on the PD dataset as one of the future work.

2) Model Comparison: In TABLE V, we see that in terms of the mean f1-score, TSA (UCF-101) outperforms I3D (UCF-101) on six tasks with a significant margin. Besides, the average mean f1-score of the former is also better than the later without any doubt. Since the only difference between the two is the backbone used, we can conclude that our TSA performs better than I3D on the PD dataset.

Also, compared to I3D (Kinetics-400), TSA (UCF-101) still has 1.5% improvements even if pre-trained from a much smaller and less complex dataset. It demonstrates that TSA is better at dealing with the large discrepancy of motion difference between non-medical datasets and our PD dataset. So we think TSA pretrained from Kinetics-400 would further improve the performance. Due to the limit of time and computation resource, we leave it as the futher work.

Regarding the time cost of the temporal relative self-attention, it is completely acceptable as the network can still run with a bit more time cost. However, the memory cost can be problematic if the network is too deep due to the hardware memory limitation. As such, we give some useful solutions in terms of the algorithm itself:

1) only replace convolutional layers with small temporal size (usually the last few),
2) reduce d_k and
3) use large kernel size or stride on the temporal dimension at the first few layers to quickly decrease the temporal size to the one you want and use kernel size of 1 at following layers to maintain the temporal size unchanged until the last layer.

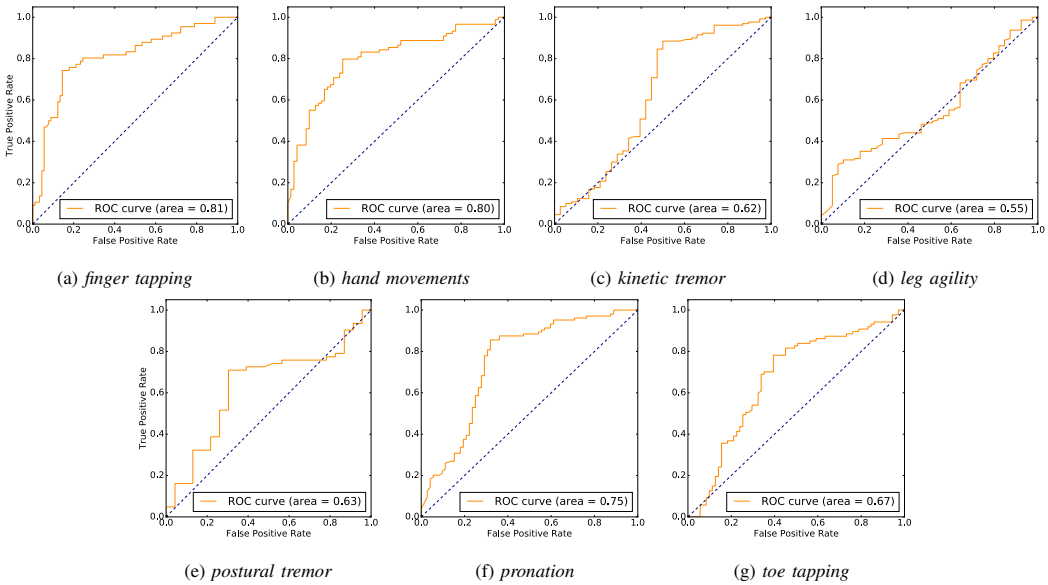
Another issue of TSA is that a large learning rate is possible to cause the exploding gradients problem, which can be overcome by applying approaches such as the ReLU activation function and pre-training.

D. Patient-level Severity Classification

We use the trained model on each task as the feature extractor to extract the learned patterns and apply the proposed

TABLE V: Accuracy, precision, recall, and f1-score on seven tasks from MDS-UPDRS-III using I3D and TSA. Datasets in the brackets denote where the model is trained. In general, I3D pre-trained from Kinetics-400 outperforms I3D trained from UCF-101, indicating transfer learning with larger datasets has more benefits than smaller datasets. TSA pre-trained from UCF-101 is already comparable to I3D pre-trained Kinetics-400, even using transfer learning with a smaller dataset.

Task	I3D (UCF-101)				I3D (Kinetics-400)				TSA (UCF-101)			
	acc	precision	recall	f1	acc	precision	recall	f1	acc	precision	recall	f1
finger	68.6	0.55	0.76	0.65	69.2	0.57	0.76	0.67	78.2	0.75	0.81	0.78
hand	70.0	0.76	0.61	0.68	77.5	0.80	0.75	0.77	75.6	0.79	0.72	0.75
kinetic	78.0	0.87	0.10	0.49	73.8	0.82	0.49	0.66	79.2	0.87	0.51	0.69
leg	79.3	0.88	0.17	0.53	79.3	0.88	0.14	0.51	70.1	0.81	0.35	0.58
postural	74.1	0.85	0.08	0.47	77.6	0.87	0.34	0.61	70.6	0.78	0.56	0.67
pronation	68.8	0.76	0.56	0.66	77.8	0.87	0.71	0.77	72.2	0.76	0.67	0.71
toe	64.6	0.72	0.52	0.62	67.7	0.70	0.65	0.68	62.0	0.68	0.53	0.61
average	-	-	-	0.59±0.09	-	-	-	0.67±0.09	-	-	-	0.68±0.07



(a) *finger tapping* (b) *hand movements* (c) *kinetic tremor* (d) *leg agility*

(e) *postural tremor* (f) *pronation* (g) *toe tapping*

Fig. 5: ROC curves for seven tasks from MDS-UPDRS-III. The ROCs on task *finger*, *hand*, *pronation* and *toe* are well shaped, indicating that models on these tasks performs well. The remaining ROCs are close to the diagonals, which means the models' performance is not good.

four task-assembling methods to incorporate tasks to produce a single concluding severity score for a patient. The patient-level severity is split into three classes by cut-off: *slight* $\in [0, 23]$, moderate $\in (23, 40]$ and *severe* $\in (40, -]$ with equal number of patients. The detailed clinical information for each class can be found in TABLE VII. Experiments are repeated 20 times to ensure validity.

1) Single-Task Baseline: To demonstrate the effectiveness of task-assembling methods, we first do patient-level severity classification using only one single task as the baseline. The result is shown in TABLE VIII. The performance rank using each task has a relative position similar to the one in task-level severity classification shown in TABLE VI. The best mean f1-score is 0.60 using single task *hand*, which is served as the baseline to compare with assembling methods.

2) Benefit from Task-assembling methods: Four task-assembling methods incorporate seven tasks used in task-level severity classification. From TABLE IX, we see that all task-assembling methods, including the most straightforward averaging strategy, outperforms the single-task baseline. The best method is the pixel-wise self-attention based weighting in terms of the accuracy and mean f1-score, with an improvement of 4.9% from the baseline. These results demonstrate that patient-level severity classification benefits from all tasks combined compared to based on a single task, which is intuitive since it is also hard for experts to diagnose a patient by inspecting just one task.

Comparing all four methods, we see the weighting strategy is better than just simple averaging, indicating that each task contributes unequally to the patient-level severity. More-

TABLE VI: The best mean f1-score on seven tasks from MDS-UPDR-III regardless of the architecture used. The rank tells us the top-3 well performed tasks are *finger*, *hand* and *pronation*.

Task	f1	Rank
finger	0.78	1
hand	0.77	2
kinetic	0.69	4
leg	0.58	7
postural	0.67	6
pronation	0.77	2
toe	0.68	5
average	**0.71**	-

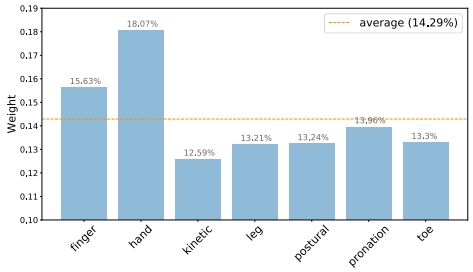


Fig. 6: Weights for seven tasks learned by vector weighting method. The weights of task *finger* and *hand* are higher than the average, which means in the task-assembling approach, i.e., vector weighting, they contribute more than other tasks in the prediction of the patient-level severity.

over, the attention-based weighting slightly outperforms the learnable vector-based weighting. It is because 1) *layer4* has more feature points, potentially more representable for a task than *avgpool*, and 2) attention-based weighting gives more flexibility to the weights such that patients can have task weights exclusively learned based on their condition.

We show the weights learned in the vector weighting method in Fig. 6 to give a general feeling of which task may contribute less or more to the prediction of patient-level severity. Weights are averaged across 20 runs on each fold, a total of 100 runs. As the two attention-based weighting methods assign task weights for patients exclusively, it is not intuitive to see the overall weight distribution on tasks. In Fig. 6, we see the top-2 tasks with highest weights are *hand* and *finger*, which well matches the performance rank in TABLE VIII. The rest tasks remain the similar position as in TABLE VIII except that task *kinetic* drops to the lowest rank. We suspect the reason being the effect of severe class imbalance problem of task *kinetic*.

3) Distinguishing between slight and severe classes: We remove the class *moderate* with the remaining classes untouched to focus on the classification between *slight* and *severe* classes. The result of the best single task baseline and assembling methods are shown in TABLE X. By combining seven tasks, we gain 0.3%-3.1% performance improvements compared to using a single task. At best, we can achieve an mean f1-score of 0.83 on distinguishing between *slight* and *severe* classes. Moreover, the attention-based weighting methods still outperform the vector-based ones, matching the case in TABLE IX.

In general, attention-based weighting strategy is the first choice to assemble the tasks, but the vector-based one is also applicable, given its higher time efficiency. It is also worthwhile to exclude some tasks to see the ablation effects on patient-level performance. As the main focus of this paper is to show the potential of combining tasks, we leave it as future work.

In Section V-D.2 and V-D.3, we empirically show the possibility that a multi-task algorithm based on an incomplete video-overview (i.e. not all MDS-UPDRS-III items are included) can help discriminate between groups of disease severity in both *slight-moderate-severe* and *slight-severe* cases with acceptable mean f1-scores, 0.63 for the former case and 0.83 for the latter case. Besides, the performance of single task and weights visualization demonstrates the tests of bradykinesia hands among all videotaped items are best reflective of the total MDS-UPDRS-III.

VI. Conclusion

In this paper, we successfully apply deep architectures on the PD video dataset to automatically identify the task-level severity, i.e., item scores in MDS-UPDRS-III given the video of the task, with satisfactory performance in terms of both accuracy and mean f1-score. Due to the small size of our PD dataset, we employ transfer learning from non-medical datasets to improve the performance of the model.

On the aspect of algorithm, we propose a new method, namely TSA, for action recognition problem and validate it on two commonly used public datasets and our Parkinson's dataset. The promising results compared to I3D demonstrate the effectiveness of TSA and better ability of handling motion discrepancy between non-medical datasets and our PD dataset during transfer learning. TSA is highly flexible which can be embedded in any 3D network for action recognition by replacing the CNN layer with the temporal relative self-attention block.

We propose four task-assembling methods to incorporate tasks to identify the patient-level severity by using the models trained on each task. Compared to using only a single task, tasks combined can produce a better performance under both two classification scenarios: *slight-moderate-severe* and *slight-severe*. It is clinically interesting that through analysis of multiple tasks, we can give a summarized severity score, which can assist in the manual rating of Parkinson's disease with reasonably good accuracy and mean f1 score.

References

[1] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International workshop on human behavior understanding*, pages 29–39. Springer, 2011.

[2] EA Belalcazar-Bolanos, JD Arias-Londono, JF Vargas-Bonilla, and JR Orozco-Arroyave. Nonlinear glottal flow features in parkinson's disease detection. In *2015 20th Symposium on Signal Processing, Images and Computer Vision (STSIVA)*, pages 1–6. IEEE, 2015.

TABLE VII: Clinical information for three classes in patient-level classification. Note that each patient is videotaped two or three times, and the severity score of each time may fall into different classes. For simplicity, L-OFF, L-ON, A, B, and C denote Levodopa challenge test OFF, Levodopa challenge test ON, Med-OFF-Stim-ON, Med-OFF-Stim-OFF, and Med-ON-Stim-ON. Each class has an approximately equal number of patients and videos, i.e., no severe class imbalance issue.

Class	Score	Number of patients	Age	Disease duration (year)	Male/Female	Number of videos					
						all	L-OFF	L-ON	A	B	C
0	15±4	32	61±8	11±4	22/10	351	0	130	66	0	155
1	32±5	32	65±9	12±5	28/4	374	62	36	145	65	66
2	53±8	31	64±8	11±5	21/10	357	152	21	12	172	0
total	33±16	39	63±8	11±5	28/11	1082	214	187	223	237	221

TABLE VIII: Single task baseline for patient-level severity classification (three classes). Rank is calculated based on the average mean f1-score from two inputs. The top-3 well-performed tasks used for patient-level classification are task *hand*, *finger* and *kinetic*. Task *hand* achieves a mean f1 score of 0.6, which is used as the best single-task baseline.

Task	Input	Accuracy	f1	Rank
finger	*avgpool*	60.3±2.8	0.58±0.05	2
	layer4	60.7±3.2	0.60±0.04	
hand	*avgpool*	61.5±2.8	0.60±0.04	1
	layer4	60.7±3.1	0.60±0.03	
kinetic	*avgpool*	59.7±2.7	0.58±0.04	2
	layer4	60.5±3.4	0.60±0.04	
leg	*avgpool*	50.6±2.7	0.45±0.05	6
	layer4	60.0±3.7	0.59±0.04	
postural	*avgpool*	54.9±2.5	0.48±0.05	5
	layer4	60.8±3.5	0.60±0.04	
pronation	*avgpool*	59.3±3.2	0.56±0.05	4
	layer4	61.3±3.4	0.60±0.04	
toe	*avgpool*	51.3±2.8	0.46±0.06	6
	layer4	60.6±3.9	0.59±0.04	

TABLE IX: Patient-level severity classification (three classes) using single task and task-assembling approaches (seven tasks). The four task-assembling methods outperform the single-task baseline with the channel-wise and pixel-wise attention weighting being the best methods.

Method	Input	Accuracy	f1
single task baseline	*avgpool*	61.5±2.8	0.60±0.04
vector averaging	*avgpool*	62.7±2.4	0.60±0.04
vector weighting	*avgpool*	64.1±2.4	0.63±0.05
channel-wise attention weighting	*layer4*	64.5±3.1	0.63±0.04
pixel-wise attention weighting	*layer4*	64.5±2.8	0.63±0.03

TABLE X: Patient-level severity classification (two classes with class *moderate* removed) using single task and task-assembling approaches (seven tasks). The four task-assembling methods outperform the single-task baseline with the pixel-wise attention weighting being the best method.

Method	Accuracy	f1
single task baseline	81.1±2.2	0.80±0.04
vector averaging	81.4±1.7	0.80±0.03
vector weighting	81.9±2.1	0.81±0.04
channel-wise attention weighting	82.2±3.1	0.82±0.05
pixel-wise attention weighting	83.6±1.2	0.83±0.02

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, 2019.

[4] Noopur A Bhalchandra, R Prashanth, Sumantra Dutta Roy, and Santosh Noronha. Early detection of parkinson's disease through shape based features from 123 i-ioflupane spect imaging. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 963–966. IEEE, 2015.

[5] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[6] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934, 2010.

[7] Kelvin L Chou, Jennifer L Taylor, and Parag G Patil. The mds- updrs tracks motor and non-motor improvement due to subthalamic nucleus deep brain stimulation in parkinson disease. *Parkinsonism & related disorders*, 19(11):966–969, 2013.

[8] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

[9] Ross Cutler and Larry S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000.

[10] Abe Davis, Katherine L Bouman, Justin G Chen, Michael Rubinstein, Fredo Durand, and William T Freeman. Visual vibrometry: Estimating material properties from small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 5335–5343, 2015.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[12] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[13] Mark Dredze and Koby Crammer. Online methods for multi-domain learning and adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 689–697, 2008.

[14] Peter Drotár, Jiří Mekyska, Irena Rektorová, Lucia Masarová, Zdeněk Smékal, and Marcos Faundez-Zanuy. Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease. *Artificial intelligence in Medicine*, 67:39–46, 2016.

[15] Luc JW Evers, Jesse H Krijthe, Marjan J Meinders, Bastiaan R Bloem, and Tom M Heskes. Measuring parkinson's disease over time: The real-world within-subject reliability of the mds-updrs. *Movement Disorders*, 34(10):1480–1487, 2019.

[16] SRLE Fahn. Unified parkinson's disease rating scale. *Recent development in Parkinson's disease*, 1987.

[17] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

[18] Christopher G Goetz, Barbara C Tilley, Stephanie R Shaftman, Glenn T

Stebbins, Stanley Fahn, Pablo Martinez-Martin, Werner Poewe, Cristina Sampaio, Matthew B Stern, Richard Dodel, et al. Movement disorder society-sponsored revision of the unified parkinson's disease rating scale (mds-updrs): scale presentation and clinimetric testing results. *Movement disorders: official journal of the Movement Disorder Society*, 23(15):2129–2170, 2008.

[19] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[20] CJ Hartmann, L Wojtecki, J Vesper, J Volkmann, SJ Groiss, A Schnitzler, and M Südmeyer. Long-term evaluation of impedance levels and clinical development in subthalamic deep brain stimulation for parkinson's disease. *Parkinsonism & related disorders*, 21(10):1247–1250, 2015.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[24] Mahesh Joshi, Mark Dredze, William Cohen, and Carolyn Rose. Multi-domain learning: when do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1302–1312, 2012.

[25] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[26] Julian FP Kooij and Jan C van Gemert. Depth-aware motion magnification. In *European Conference on Computer Vision*, pages 467–482. Springer, 2016.

[27] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011.

[28] Franziska Maier, Catharine J Lewis, Nina Horstkoetter, Carsten Eggers, Till A Dembek, Veerle Visser-Vandewalle, Jens Kuhn, Mateusz Zurowski, Elena Moro, Christiane Woopen, et al. Subjective perceived outcome of subthalamic deep brain stimulation in parkinson's disease one year after surgery. *Parkinsonism & related disorders*, 24:41–47, 2016.

[29] Franziska Maier, Catharine J Lewis, Nina Horstkoetter, Carsten Eggers, Elke Kalbe, Mohammad Maarouf, Jens Kuhn, Mateusz Zurowski, Elena Moro, Christiane Woopen, et al. Patients' expectations of deep brain stimulation, and subjective perceived outcome related to clinical measures in parkinson's disease: a mixed-method approach. *Journal of Neurology, Neurosurgery & Psychiatry*, 84(11):1273–1281, 2013.

[30] Attila Makkos, Márton Kovács, Zsuzsanna Aschermann, Márk Harmat, József Janszky, Kázmér Karádi, and Norbert Kovács. Are the mds-updrs–based composite scores clinically applicable? *Movement Disorders*, 33(5):835–839, 2018.

[31] Pablo Martinez-Martin, Carmen Rodriguez-Blazquez, Mario Alvarez-Sanchez, Tomoko Arakaki, Alberto Bergareche-Yarza, Anabel Chade, Nelida Garretto, Oscar Gershanik, Monica M Kurtis, Juan Carlos Martinez-Castrillo, et al. Expanded and independent validation of the movement disorder society–unified parkinson's disease rating scale (mds-updrs). *Journal of neurology*, 260(1):228–236, 2013.

[32] Marcelo Merello, Eliana Roldan Gerschcovich, Diego Ballesteros, and Daniel Cerquetti. Correlation between the movement disorders society unified parkinson's disease rating scale (mds-updrs) and the unified parkinson's disease rating scale (updrs) during l-dopa acute challenge. *Parkinsonism & Related Disorders*, 17(9):705–707, 2011.

[33] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[34] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[35] Silvia L Pintea, Jian Zheng, Xilin Li, Paulina JM Bank, Jacobus J van Hilten, and Jan C van Gemert. Hand-tremor frequency estimation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[36] Ramprasad Polana and Randal C Nelson. Detection and recognition of periodic, nonrigid motion. *International Journal of Computer Vision*, 23(3):261–282, 1997.

[37] Michalis Raptis and Leonid Sigal. Poselet key-framing: A model for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2650–2657, 2013.

[38] Gerard Saranza and Anthony E Lang. Levodopa challenge test: indications, protocol, and guide. *Journal of neurology*, 2020.

[39] Scott Satkin and Martial Hebert. Modeling the temporal extent of actions. In *European conference on computer vision*, pages 536–548. Springer, 2010.

[40] Sebastian Schade, Friederike Sixel-Döring, Jens Ebentheuer, Xenia Schulz, Claudia Trenkwalder, and Brit Mollenhauer. Acute levodopa challenge test in patients with de novo parkinson's disease: data from the denopa cohort. *Movement disorders clinical practice*, 4(5):755–762, 2017.

[41] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[42] David G Standaert, Marie H Saint-Hilaire, and Cathi A Thomas. *Parkinson's Disease Handbook*. American Parksinon Disease Association, New York, USA, 2015.

[43] Renfei Sun, Zhiyong Wang, Kaylena Ehgoetz Martens, and Simon Lewis. Convolutional 3d attention network for video based freezing of gait recognition. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7. IEEE, 2018.

[44] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[45] Sigurlaug Sveinbjornsdottir. The clinical symptoms of parkinson's disease. *Journal of neurochemistry*, 139:318–324, 2016.

[46] Chusak Thanawattano, Chanawat Anan, Ronachai Pongthornseri, Songphon Dumnin, and Roongroj Bhidayasiri. Temporal fluctuation analysis of tremor signal in parkinson's disease and essential tremor subjects. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6054–6057. IEEE, 2015.

[47] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[48] Travis H Turner and Marian L Dale. Inconsistent movement disorders society–unified parkinson's disease rating scale part iii ratings in the parkinson's progression marker initiative. *Movement Disorders*, 2020.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[50] Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T Freeman. Phase-based video motion processing. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[51] Ferdous Wahid, Rezaul K Begg, Chris J Hass, Saman Halgamuge, and David C Ackland. Classification of parkinson's disease gait using spatial-temporal gait features. *IEEE journal of biomedical and health informatics*, 19(6):1794–1802, 2015.

[52] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[53] Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1(08), 2007.

[54] David C Wong, Samuel D Relton, Hui Fang, Rami Qhawaji, Christopher D Graham, Jane Alty, and Stefan Williams. Supervised classification of bradykinesia for parkinson's disease diagnosis from smartphone videos. In *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 32–37. IEEE, 2019.

[55] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

2

INTRODUCTION

Parkinson's disease (PD) is mainly diagnosed, in typical cases, based on symptoms (with tests like neuroimaging), which can be recognized by human experts. Recently, the usage of sensor devices attached to the patients and machine/deep learning based methods arise to help automatically detect the symptom and its severity [2, 9, 16] as an objective assistant to the human experts. These machine learning methods try to detect the distinctive features typically from four kinds of data: signal dynamics (e.g, handwriting [27] and motor data from wearable device [28]), vocal data [4], images (e.g, magnetic resonance images [33]) and videos (e.g, clinical videos of freezing of gait [31]). In this study, we focus on automatic PD severity classification from videos using deep learning.

2.1. MOTIVATION

AMONG the data types used in machine learning approaches, video is not yet fully explored to show the potential for PD diagnosis. Besides, most methods based on clinical videos extract hand-crafted features from videos as the input data rather than directly utilizing the videos as the input and thus problem dependent and time-consuming. We wonder if it is possible to do PD severity classification using the videos as the input data. In [31], researchers show the possibility of PD classification from the freezing of gait videos using 3D CNN, which gives us the confidence to explore this task further. Only one type of clinical task is used in [31], and one task is usually not sufficient for an accurate diagnosis for patients. Therefore, we are still curious if videos from other tasks (such as the tasks in MDS-UPDRS-III) can also be used as the input for PD severity classification with acceptable accuracy.

2.2. RESEARCH QUESTIONS

THE first research question is:

- Can we perform automatic PD severity classification from videos of multiple tasks in MDS-UPDRS-III using deep architectures?

As the size of the clinical dataset we use is small, it is hard to train the network on the dataset from scratch efficiently. Thus we try to use transfer learning to pre-train the network on large public datasets to overcome this issue. Then the second research question can be:

- Can the network benefit from public video datasets using transfer learning?

Comparing public video datasets with the clinical dataset, we can observe that there are substantial motion differences among classes in public datasets. In the clinical dataset, however, the motion difference is too subtle to be recognized by inexperienced severity raters. This discrepancy between source and target dataset can reduce the positive effect of transfer learning and increase its difficulty to be applied. So the third research question can be:

- How to overcome the issue that brings by the large discrepancy between source and target domain during transfer learning?

Here, the PD severity classification is only performed on multiple tasks separately from each patient, and thus each patient has separate severity for each task. However, it is more clinically interesting to produce a summary of the patient's severity rather than multiple ones. So we form the forth as well as the final research question as:

- How to combine the trained model on different tasks to produce a final summary severity score for the patient?

3

Background on Deep Learning

Deep learning is a class of machine learning methods based on artificial neural networks with representation learning. It allows complex computational models to learn and represent the data with multiple levels of abstraction. Different from conventional machine learning techniques, deep learning has the advantage that there is no longer the need to carefully design a feature extractor to transform raw data to feature representation based on which a classification model can make predictions. Therefore, deep learning can be a powerful and flexible tool for dealing with problems or data from which informative features cannot be extracted quickly and easily.

In computer vision, deep learning has facilitated solving various problems, such as object detection, motion tracking, action recognition, and human pose estimation semantic segmentation [14, 17, 19, 29, 32]. All these tasks are solved with representation-learning methods, mostly based on Convolutional Neural Networks (CNNs) inspired by the visual system's structure. It is the combination of CNNs and deep learning that accelerates the improvement of tackling computer vision tasks. In this chapter, we introduce the basic idea of CNNs and parts of our proposed network.

3.1. Convolutional Neural Networks

The key idea of CNNs is to replace matrix multiplication in neural nets with convolution and leave everything else untouched such as maximum likelihood, back-propagation. CNN assumes the input is laid out on a grid (1-D, 2-D, 3-D). Mostly, a CNN comprises three main types of neural layers, i.e., convolutional layers, non-linearity Layers, and pooling layers, with each layer having a specific function that transforms input to output with or without learnable parameters. A demonstration of the overall structure is shown in Figure 3.1. We explain each component in the following sections.

3.1.1. Convolutional Layer

In CNNs, the convolutional layer is the core building block that takes up most of the computations, and its task is to detect local patterns from the previous layer and map

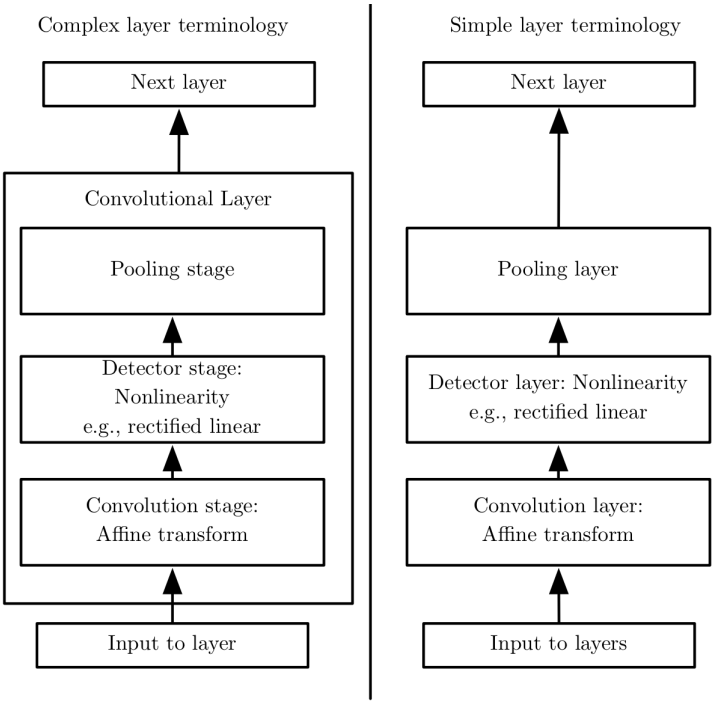


Figure 3.1: CNN components [12] including convolutional layer, non-linearity layer and pooling layer.

the patterns to a feature map.

A convolutional layer contains a set of learnable filters of size *width* (W) × *height* (H) × *depth* (D) (2-D convolution) and each filter connects to only a local region of the input volume. The convolution is operated between the filter and the input connected to that filter. An illustration of the convolutional layer is shown in Figure 3.2. Each filter has the same depth as the input volume, and the number of filters applied equals the depth of the output volume. The output volume is computed by element-wise multiplying the input volume with each filter and summing up the results from all filters, with bias offset. Formally, the process of calculating each element of the output volume can be formulated as

$$X_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{D^{(l-1)}} K_{i,j}^{(l)} * X_j^{(l-1)}, \tag{3.1}$$

where $X_i^{(l)}$ is the i^{th} feature map of output $(W^{(l)} \times H^{(l)} \times D^{(l)})$ of layer l; $K_{i,j}^{(l)}$ is the filter of size $F_W \times F_H$ connecting the j^{th} feature map in layer $l-1$ with i^{th} feature map in layer l, and $b_i^{(l)}$ is the bias matrix.

As each filter connects to only a local region of the previous layer's neuron, which leads to a sparse connection, the number of parameters in convolution can be vastly reduced compared to dense connection. The comparison between sparse and dense con-

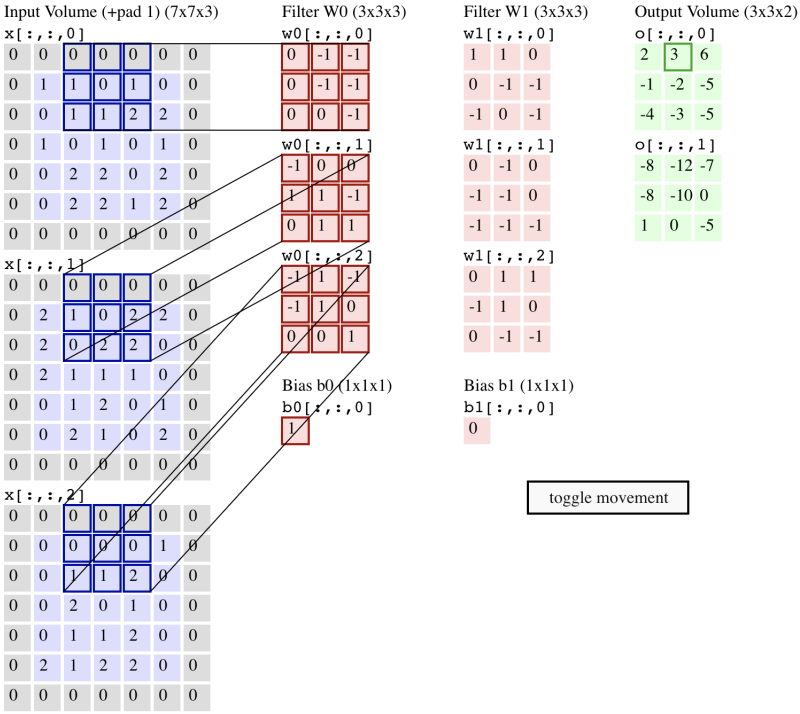


Figure 3.2: Illustration of convolution with each depth slice of input volumes, filters and output volumes visualized in row stack [18]. Element-wise multiplication is performed between input regions and filters. The target value in output volume is the summation of multiplication's outputs. The number of filters used decides how many channels in the output layer.

nection is shown in Figure 3.3. Besides, the parameters of each filter are shared across all spatial locations. These two properties make it possible to scale up neural networks to process large images or video sequences.

3.1.2. NON-LINEARITY LAYER

The neural network (NNs) can be seen as a complex function mapping input to output with tolerable accuracy. However, without non-linearity, the complex NN is only able to learn a function similar to the linear model, even keeping adding layers to the network, not to say learning patterns automatically from complex inputs. To make the NNs/CNNs capable of learning complex structures, we need to add non-linearity function to the network, called non-linear activation function in the non-linearity layer.

The activation function takes the feature map generated by the convolutional layer and performs activation operation on the feature map to output the activation map. As the activation function applies element-wise operation on the feature map, the shape of the activation function's input and output is the same. The operation of activation

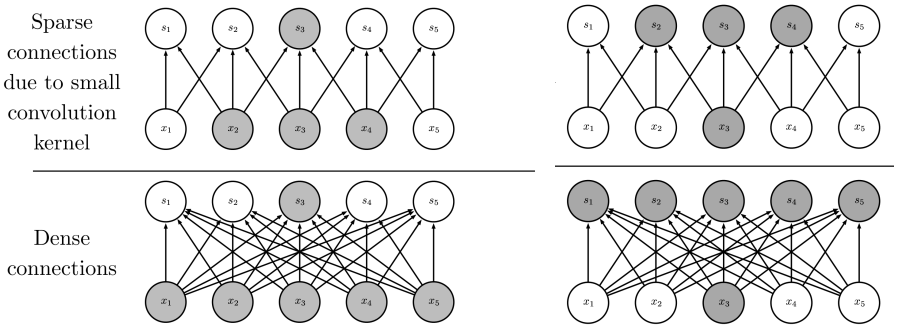


Figure 3.3: Comparison between sparse and dense connection [12]. In the upper figure, neurons are only connected to its neighbors, which is the sparse connection. In the lower figure, neurons are connected to every other neuron, which is the dense connection.

function can be formulated as

$$X^{(l)} = \sigma(X^{(l-1)}), \tag{3.2}$$

with

$$\begin{aligned} &X^{(l)} \in \mathbb{R}^{W^{(l)} \times H^{(l)} \times D^{(l)}}, \\ &X^{(l-1)} \in \mathbb{R}^{W^{(l-1)} \times H^{(l-1)} \times D^{(l-1)}}, \\ &W^{(l)} = W^{(l-1)}, H^{(l)} = H^{(l-1)}, D^{(l)} = D^{(l-1)}, \end{aligned} \tag{3.3}$$

where σ is the activation function.

There are four commonly used non-linear activation functions: 1) tanh

$$x^{(l)} = \tanh(x^{(l-1)}), \tag{3.4}$$

2) sigmoid

$$x^{(l)} = \frac{e^{x^{(l-1)}}}{1 + e^{x^{(l-1)}}}, \tag{3.5}$$

3) ReLU [24]

$$x^{(l)} = \max(0, x^{(l-1)}) \tag{3.6}$$

and 4) Leaky ReLU

$$x^{(l)} = \begin{cases} 0.01x^{(l-1)} & \text{if } x^{(l-1)} < 0 \\ x^{(l-1)} & \text{otherwise,} \end{cases} \tag{3.7}$$

where x denotes a single element of X. The most commonly used activation function in the neural network is ReLU because of its reduced likelihood of vanishing gradient, increased probability of resulting sparse representation, and computational efficiency.

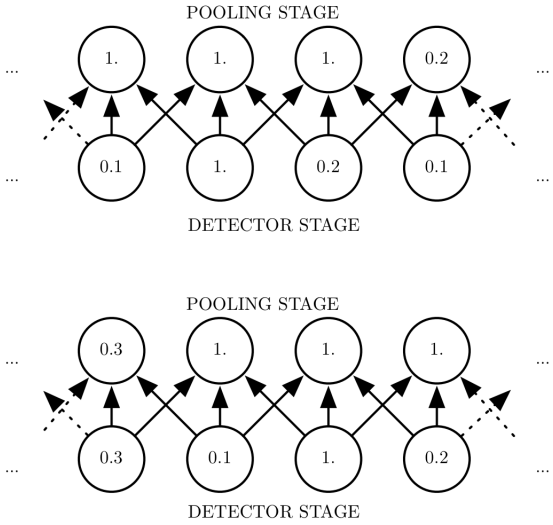


Figure 3.4: The max pooling provides approximately translation invariance [12]. Compared to the top subfigure, the inputs of the bottom one are shifted one pixel right. However, the outputs of the pooling layer do not change much because max pooling is only sensitive to the maximum value in the window region instead of its absolute position.

3.1.3. POOLING LAYER

The pooling layer is usually applied after multiple stages of convolutional layers or activation layers to reduce the spatial size of the feature map or activation map as well as minimizing the likelihood of overfitting.

The pooling layer has only two parameters, the spatial extent $F^{(l)}$ and stride $S^{(l)}$. It takes the input volume of size $W^{(l-1)} \times H^{(l-1)} \times D^{(l-1)}$ and generates output volume of size $W^{(l)} \times H^{(l)} \times D^{(l)}$ where

$$\begin{aligned} W^{(l)} &= \frac{(W^{(l-1)} - F^{(l)})}{S^{(l)}} + 1, \\ H^{(l)} &= \frac{(H^{(l-1)} - F^{(l)})}{S^{(l)}} + 1, \\ D^{(l)} &= D^{(l-1)}. \end{aligned} \tag{3.8}$$

The pooling layer's key idea is to ensure approximate translation invariance in vision tasks where it is more important to detect the patterns instead of locating the patterns. See Figure 3.4 for the illustration of translation invariance. The pooling layer manages to keep the most prominent patterns or features detected from previous layers in a smaller representation by discarding less prominent ones with the loss of spatial resolution.

The most commonly used pooling methods are max pooling and average pooling. The former finds the max value in the pooling window while the latter finds the average value. Because of faster convergence and better performance, max pooling is mostly used in state of the art deep architectures.

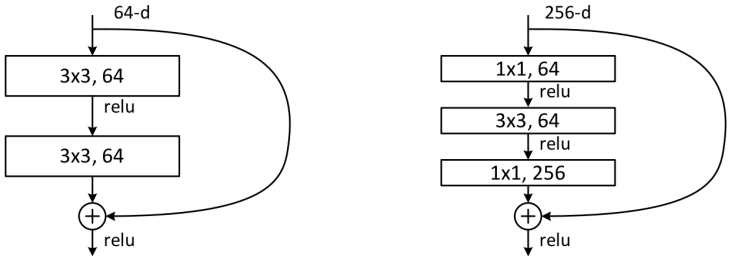


Figure 3.5: ResNet blocks [14]. Left one is the basic block and right one is the bottleneck block. The basic block has two convolutional layers with 3 by 3 filters. The bottleneck block has one convolutional layer with 3 by 3 filter and two convolutional layers with 3 by 3 filters.

3.2. RESNET

RESNET [14] is a class of CNNs which significantly improved the performance on image classification and object detection tasks. ResNet manages to learn residuals from the layer inputs and is demonstrated to be easier to optimize, making it possible to train substantially deep neural networks. In [14], five variants of ResNet are introduced: ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, with the number indicating how many layers adopted in the networks. The core building blocks for ResNet are the basic block for ResNet-18/34, and bottleneck block for ResNet-50/101/152. The two blocks are shown in Figure 3.5.

3.3. SELF-ATTENTION

SELF-ATTENTION is a class of attention mechanism, and it can capture the interactions within the sequence of inputs. Like attention, self-attention can also assign different importance to the inputs, allowing the network to learn which part of the inputs should be focused more. Self-attention is heavily explored since the Transformer network [37] was proposed for machine translation tasks where the self-attention is used to compute the interactions between words in the sentence.

In image tasks, self-attention is usually used to compute the interactions among pixels in the same image and allows the model to learn which part of the image is of more importance [3]. In video tasks, we can use self-attention to learn the long-range interactions among voxels in the same video [38] though [38] denotes the self-attention as the non-local block. As a video contains both spatial and temporal dimensions and its size is usually much larger than a single image, directly computing interactions among all voxels is both time and memory inefficient. Thus in this work, we use self-attention solely on the temporal dimension to capture the interactions through time and leave CNN to learn the features on the spatial dimension.

The core matrices of self-attention are the query, key, and value matrix, which are learned through linear transformation, e.g., convolution operation. The self-attention maps a query and a set of key-value pairs to an output. The output is the weighted sum of the values, and the weights are produced from the incorporation (i.e., operations between matrices) of the query and key. Figure 3.6 shows the overall procedure

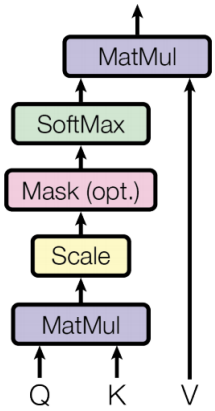


Figure 3.6: The building block of self-attention [37]. Dot products are computed among query Q and keys Ks. The products are divided by a scaling factor to avoid diminishing gradients in softmax layer. The softmax is applied to obtain the weights for the values. Finally, the weights and values are multiplied to form the final outputs of self-attention.

of the self-attention block. The query and key have a dimension of d_k and the value of d_v. First, the dot products are computed among queries and all keys. The products are further divided by a scaling factor $\sqrt{d_k}$ to counteract the effect that dot products can grow extremely large for large d_k, causing diminishing gradients in the softmax layer. The softmax function is applied to the outputs of dot products to obtain the weights for the values. The whole procedure can be divided into two parts: computing weights for values and computing outputs of self-attention by multiplying the weights with values. The weights produced by the key and query can represent the importance of different parts of the inputs. In the image, it means which pixel is more important, and in the video, it means which voxel is more critical. Therefore, one can also see self-attention as an approach to assign different importance on the inputs, enabling the model to learn the most related patterns.

3.4. RELATIVE POSITIONAL EMBEDDINGS

As self-attention is computed among all elements in input, e.g., the pixel in an image, and it is permutation equivalent about the position of the elements, e.g., the outputs do not change if we change the position of the pixel in the image. This property can affect the effectiveness of modeling structured data, where the position is a type of crucial information. For instance, if we change the pixels' positions in an image where there is a cat, the model or even ourselves can not recognize the cat well. We obtain global interactions while losing the local ones. The relative positional embeddings (E^r) are employed to prevent permutation equivariance and allow attention to know how far two positions are apart in the input.

For a 1D input, the E^r the learnable embeddings are of shape $H \times L \times d_k$, where H, L and d_k are the number of heads, sequence length and the dimension of key. E^r has an

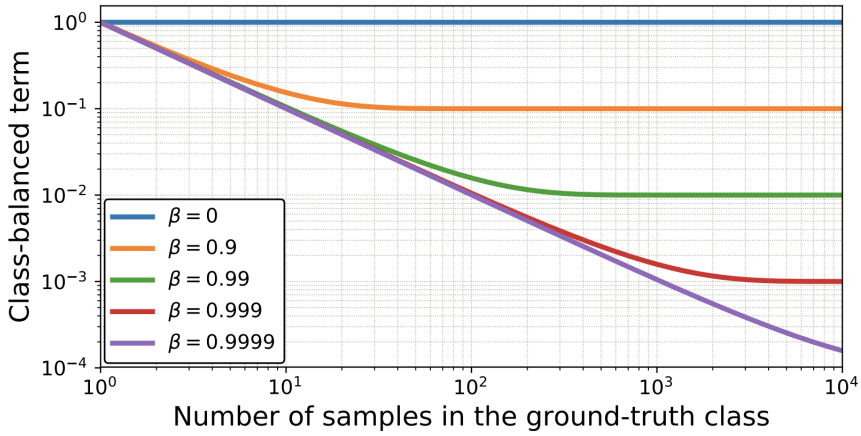


Figure 3.7: Caption

embedding for each pairwise distance $r = j_k - i_q$ between a query and key in position i_q and j_k. First, we need to compute the intermediate tensor R of shape $L \times L \times d_k$, which contains the embeddings between all keys and queries in relative distances. Then we multiply R with reshaped query matrix Q to get the $L \times L$ logits matrix S^{rel}. With self-attention, the output of relative self-attention is formed as:

$$O = \text{Softmax}\left(\frac{QK^T + S^{rel}}{\sqrt{d_k}}\right)V. \tag{3.9}$$

The relative positional embeddings can complement the drawback of position-unaware of self-attention.

3.5. CLASS-BALANCED LOSS

The class-balanced loss is used to solve the training problem on imbalanced data by adding a weighting factor that is inversely proportional to the effective number of samples [6] E_{n_i}. $E_{n_i} = (1-\beta_i^{n_i})/(1-\beta_i)$ for class i, where n_i is the number of samples in class i and $\beta \in [0,1)$ is the smoothing factor. Suppose the original loss is L, the class-balanced (CB) loss is formed as:

$$L_{\text{CB}} = \frac{1-\beta}{1-\beta^{n_y}}L, \tag{3.10}$$

where n_y is the number of samples in the ground-truth class y. Figure 3.7 visualize the class-balanced loss as a function of n_y for different β. In general, small weights are assigned to the class with large number of samples. When increasing β from 0 to 1 (exclusive), the effect of re-weighting increases.

4

Action Recognition

When it comes to CNNs, our first thought is its great power in solving vision tasks (e.g., detection, tracking, segmentation) from image data. Indeed, researches have made significant progress in solving problems on images by utilizing the advantages of CNNs. Besides image tasks, video tasks have also generally become the focus of research with the appearance of vast video resources. Compared to the image, video as structured data as well usually contains more information such as motion, the interaction between objects. Action recognition is a video-based task, and it is usually more complicated than the image classification problem. For instance, in Figure 4.1, which is a single frame taken from a video, to form an action recognition task, we need to answer which action is performed in this video by observing the type of objects within it and the interactions between objects through time. The action will be hard to detect when too many objects appear, and the interactions between objects are complicated. This difficulty is all from the additional temporal dimension of videos compared to images.

In classical computer vision, the Harris-3D detector and the Cuboid detector are mostly used to detect the space-time salient points, but they depend on hand-crafted features, which is highly problem dependent. Therefore, researchers start to focus on learning low- to high-level features through representation learning or specifically deep learning to solve action recognition tasks.

4.1. Inflated 3D Convolutional Neural Network

The video can be seen as a stack of images ordered in time. Therefore most of the researches continue to use CNN as the building block for developing the deep architectures to solve video tasks.

One of the most successful deep architectures is the inflated 3D convolutional neural network (I3D) introduced by [5]. It takes the advantages of successful image classification deep architectures that have evolved over the last few years. The key concept of I3D is to reuse parameters of those architectures that have already been well trained on large-scale image datasets such as ImageNet [8].

Figure 4.1: Demo of action recognition, *Horse race* class in UCF-101 [30]. It is a screenshot of the original video.

Before moving on to the details of I3D, let us first get familiar with some popular architectures, a subset of models for video tasks. Figure 4.2 shows commonly used five kinds of models for video classification. The architecture of I3D is based on model *b*, applying convolution operation on both spatial and temporal dimensions simultaneously by utilizing convolutional layers with 3D kernels. A simple demo for 3D convolution is shown in Figure 4.3. The difference from model *b*, apart from the intrinsic structure, is that the 3D filters of I3D are inflated from 2D filters by attaching an additional temporal dimension. For example, a 2D filter of size $F_W \times F_H$ can be inflated to a 3D filter of size $F_W \times F_H \times F_T$ where W, H and T denote the width, height, and temporal dimension. In practice, we can bootstrap 3D filters from 2D filters by repeating the weights of 2D filters F_T times and dividing by F_T for rescaling to ensure that the convolutional filter response is the same. The advantage of I3D is that the weights of the 2D filters pre-trained on large dataset such as ImageNet can be copied onto 2D filters that are repeated to form 3D filters. In other words, the inflated 3D CNN is implicitly pre-trained on the image dataset, reducing the likelihood of overfitting and improving performance.

I3D is a robust type of video architecture in recent years, and it is flexible to incorporate into many CNN architectures. For example, the popular Inception family [32] and ResNet family [14] can be easily inflated into the 3D version for video tasks by bootstrapping their 2D pre-trained filters into 3D ones.

Regarding the effect of depth of I3D on the performance, it is found that with sufficient data, I3D with more layers can achieve better accuracy on the action classification task [13]. In [13], ResNet is used as the backbone network to inflate, and all versions of ResNet with various number of layers are studied to achieve state of the art performance.

4.2. SELF-ATTENTION REPLACING 3D CONVOLUTIONAL LAYER

THE 3D convolutional layer can be replaced with the temporal self-attention block to construct the temporal self-attention network for action recognition. In this net-

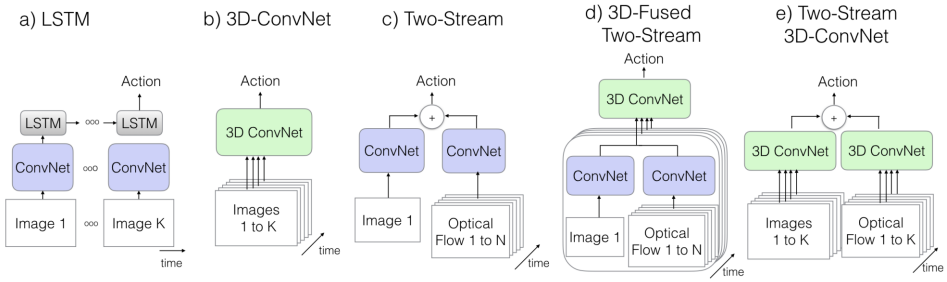


Figure 4.2: Five commonly used video architectures [5]. K and N denote the number of frames and a subset of neighboring frames in the video. Model *a* applies CNN on each frame and captures the temporal information using LSTM on the top of CNN. Model *b* applies 3D CNN directly on the video volume by performing convolution on temporal and spatial dimensions simultaneously. Model *c* combines the results from 2D convolution on one image and a stack of optical flows. Model *d* fuses results from model *b* and model *c*. Model *e* is a well-known two-stream network that combines 3D CNN results on both stacks of images and optical flows. These architectures differ in the aspects of whether convolutional layer uses 2D or 3D kernels, which kind of inputs is used such as RGB frames and optical flow, and how features are incorporated across the frames.

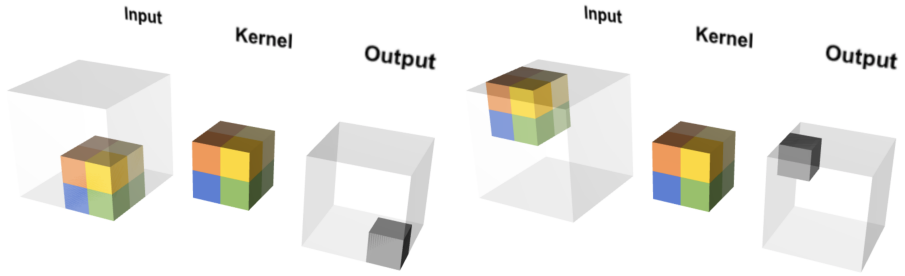


Figure 4.3: Demo for 3D convolution [1]. Similar to 2D convolution, element-wise multiplication is performed between the input regions and filters and the result is summed up to produce the value in the output volume. The difference from 2D is that 3D convolution involves the temporal dimension.

work, temporal self-attention is used to learn the long-range interactions among pixels with the same spatial position along the temporal dimension. The key idea is to explore the informative temporal patterns in the global scope to complement the shortage of CNN that distant patterns cannot interact at the same layer.

5

TRANSFER LEARNING

Transfer learning is widely used to share the knowledge learned from one task with another task, aiming to avoid expensive data collection and data labeling efforts, thus benefiting the training of models on the new tasks. When tasks are similar in terms of their feature space or distribution, transfer learning usually show a positive effect on solving the target tasks.

In action recognition, transfer learning is also frequently applied. For instance, the knowledge learned from in a larger action recognition dataset Kinetics-400 is transferred to recognizing actions on much smaller datasets UCF-101 and HMDB-51 [5, 13], where significant performance improvement is gained.

In this chapter, we briefly introduce the transfer learning and describe how it is used in this work.

5.1. BRIEF INTRODUCTION TO TRANSFER LEARNING

FOLLOWING [26], we describe some notations and then give the definition of transfer learning. Given a domain $D = \{X, P(X)\}$ where X is the feature space and $P(X)$ is the marginal probability distribution, a task can be denoted as $T = \{Y, f(\cdot)\}$. Y is the task objective, e.g., labels in classification tasks, and $f(\cdot)$ is the objective predictive function that is learned from the training data. Each sample in the data can be represented as a pair of the feature and corresponding objective $\{x_i \in X, y_i \in Y\}$. From probabilistic viewpoint, $f(\cdot)$ can also be interpreted as conditional probability distribution $P(y|x)$.

For simplicity, we consider only two domains involved in the transfer learning process, source domain D_S and target domain D_T, which is the most common case in recent research. Then the transfer learning can be defined as follows. Given a source domain D_S, source task T_S, target domain D_T, and target task T_T, the transfer learning helps improve the learning of target objective predictive function $f_T(\cdot)$ in D_T and T_T, using the knowledge learned in D_S and T_S where $D_S \neq D_T$ or $T_S \neq T_T$. Figure 5.1 shows an example of the transfer learning process.

The transfer learning taxonomy is summarized in Figure 5.2. Transfer learning is categorized into three classes: inductive transfer learning, transductive transfer learning,

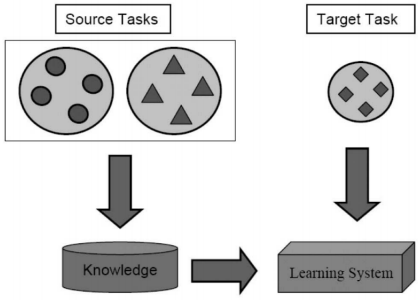


Figure 5.1: The demo of the transfer learning process. The knowledge is first learned from sourse tasks and then transfered to the learning system that tries to solve the target task.

and unsupervised transfer learning based on 1) what to transfer, 2) how to transfer, and 3) when to transfer [26].

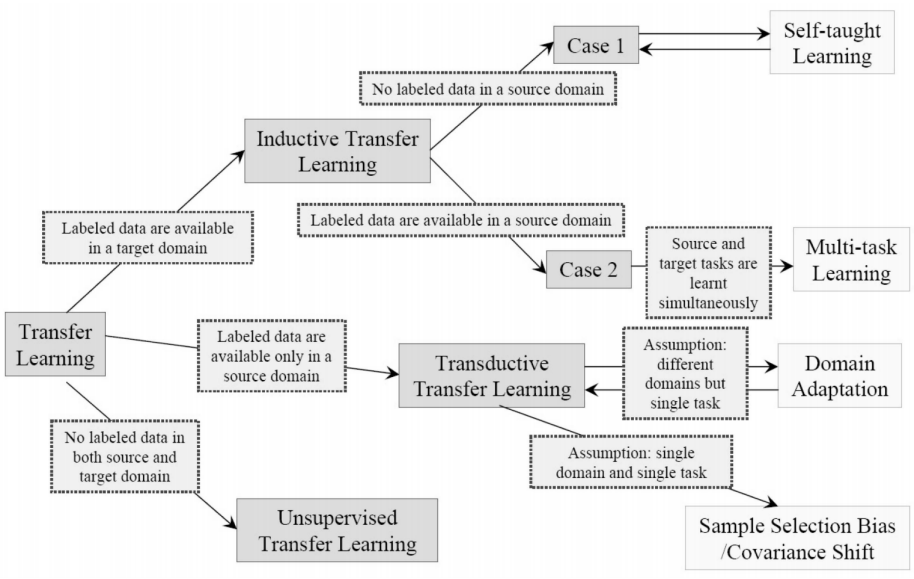


Figure 5.2: The overview of transfer learning with different settings [26]. Transfer can be categorized into three classes: inductive transfer learning, transductive transfer learning and unsupervised transfer learning.

5.2. DEEP TRANSFER LEARNING

SINCE deep neural networks dominate the machine learning field in recent years, transfer learning is intensively incorporated with deep learning. The objective predictive function $f(\cdot)$ stands for the universal function of deep neural networks. Compared to transfer learning, deep transfer learning can be classified, using a different taxonomy,

into four classes [34] based on the deep learning techniques: instances-based [7, 39], mapping-based [20, 36], network-based [15, 21, 25], and adversarial-based deep transfer learning [35]. Instances-based deep transfer learning adds part of instances (weighted) from the source domain to the target domain's training data. Mapping-based deep transfer learning maps instances in source and target domain to new data space for the target task. Network-based deep transfer learning reuses the pre-trained partial neural network (source domain) on the target domain, as a plug-and-play transferring strategy. Adversarial-based deep transfer learning uses adversarial strategies to find the feature representations that have good transferability between source and target domains. An adversarial network can be applied to distinguish representations of the source and the target domain in practice. The worse the adversarial network performs, the better transferability of the representations.

In this study, we adopt the most straightforward strategy - network-based deep transfer learning as some researches have already illustrated its effectiveness [5, 13].

6

PARKINSON'S DISEASE

This chapter briefly describes the MDS-Unified Parkinson's Disease Rating Scale, used as a criterion to label the severity score.

6.1. MDS-UNIFIED PARKINSON'S DISEASE RATING SCALE

At present, the Movement Disorder Society — Unified Parkinson's Disease Rating Scale (MDS-UPDRS), containing four parts: I for non-motor experiences of daily living, II for motor experiences of daily living, III for motor examination and IV for motor complications, has been widely used as a validated tool to quantify PD severity [11, 22]. MDS-UPDRS is the revised and more comprehensive version of the original UPDRS [10], and they are highly correlated with the motor sections [23]. This study uses the MDS-UPDRS part III (MDS-UPDRS-III) as the measurement for analysis, which contains 18 tasks and 33 scores, with some tasks pertaining to either left or right extremities. Each task, tied to a symptom assessed by clinically trained raters, has five responses linked to symptom-severity: 0-normal, 1-slight, 2-mild, 3-moderate, and 4-severe, providing consistency across tasks.

BIBLIOGRAPHY

[1] *3D convolution animation by Thom Lane*. https://thomelane.github.io/convolutions/3DConv.html. Accessed: 2020-07-30.

[2] Giovanni Albani et al. "An integrated multi-sensor approach for the remote monitoring of parkinson's disease". In: *Sensors* 19.21 (2019), p. 4764.

[3] Irwan Bello et al. "Attention augmented convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3286–3295.

[4] Abdullah Caliskan et al. "Diagnosis of the parkinson disease by using deep neural network classifier". In: *Istanbul University-Journal of Electrical & Electronics Engineering* 17.2 (2017), pp. 3311–3318.

[5] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[6] Yin Cui et al. "Class-balanced loss based on effective number of samples". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9268–9277.

[7] Wenyuan Dai et al. "Boosting for transfer learning". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 193–200.

[8] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[9] M Kelley Erb et al. "mHealth and wearable technology should replace motor diaries to track motor fluctuations in Parkinson's disease". In: *NPJ digital medicine* 3.1 (2020), pp. 1–10.

[10] SRLE Fahn. "Unified Parkinson's disease rating scale". In: *Recent development in Parkinson's disease* (1987).

[11] Christopher G Goetz et al. "Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): scale presentation and clinimetric testing results". In: *Movement disorders: official journal of the Movement Disorder Society* 23.15 (2008), pp. 2129–2170.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[13] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018, pp. 6546–6555.

[14] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[15] Jui-Ting Huang et al. “Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 7304–7308.

[16] C Kotsavasiloglou et al. “Machine learning-based classification of simple drawing movements in Parkinson’s disease”. In: *Biomedical Signal Processing and Control* 31 (2017), pp. 174–180.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[18] *Lecture of CS231n Convolutional Neural Networks for Visual Recognition*. https://cs231n.github.io/convolutional-networks/. Accessed: 2020-07-29.

[19] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[20] Mingsheng Long et al. “Learning transferable features with deep adaptation networks”. In: *International conference on machine learning*. 2015, pp. 97–105.

[21] Mingsheng Long et al. “Unsupervised domain adaptation with residual transfer networks”. In: *Advances in neural information processing systems*. 2016, pp. 136–144.

[22] Pablo Martinez-Martin et al. “Expanded and independent validation of the Movement Disorder Society–Unified Parkinson’s disease rating scale (MDS-UPDRS)”. In: *Journal of neurology* 260.1 (2013), pp. 228–236.

[23] Marcelo Merello et al. “Correlation between the Movement Disorders Society Unified Parkinson’s Disease rating scale (MDS-UPDRS) and the Unified Parkinson’s Disease rating scale (UPDRS) during L-dopa acute challenge”. In: *Parkinsonism & Related Disorders* 17.9 (2011), pp. 705–707.

[24] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *ICML*. 2010.

[25] Maxime Oquab et al. “Learning and transferring mid-level image representations using convolutional neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1717–1724.

[26] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[27] Clayton R Pereira et al. “Deep learning-aided Parkinson’s disease diagnosis from handwritten dynamics”. In: *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Ieee. 2016, pp. 340–346.

[28] Erika Rovini et al. “Comparative motor pre-clinical assessment in Parkinson’s disease using supervised machine learning approaches”. In: *Annals of biomedical engineering* 46.12 (2018), pp. 2057–2068.

[29] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[30] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *arXiv preprint arXiv:1212.0402* (2012).

[31] Renfei Sun et al. “Convolutional 3D attention network for video based freezing of gait recognition”. In: *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE. 2018, pp. 1–7.

[32] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[33] Athanasios Tagaris, Dimitrios Kollias, and Andreas Stafylopatis. “Assessment of Parkinson’s disease based on deep neural networks”. In: *International Conference on Engineering Applications of Neural Networks*. Springer. 2017, pp. 391–403.

[34] Chuanqi Tan et al. “A survey on deep transfer learning”. In: *International conference on artificial neural networks*. Springer. 2018, pp. 270–279.

[35] Eric Tzeng et al. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[36] Eric Tzeng et al. “Deep domain confusion: Maximizing for domain invariance”. In: *arXiv preprint arXiv:1412.3474* (2014).

[37] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[38] Xiaolong Wang et al. “Non-local neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.

[39] Yi Yao and Gianfranco Doretto. “Boosting for transfer learning with multiple sources”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1855–1862.