Dynamic Spatio-Temporal Specialization Learning for Fine-Grained Action Recognition

Tianjiao Li[1⋆], Lin Geng Foo[1⋆], Qiuhong Ke[2], Hossein Rahmani[3], Anran Wang[4], Jinghua Wang[5], and Jun Liu[1⋆⋆]

[1] ISTD Pillar, Singapore University of Technology and Design
{tianjiao_li,lingeng_foo}@mymail.sutd.edu.sg, jun_liu@sutd.edu.sg
[2] Department of Data Science & AI, Monash University
qiuhong.ke@monash.edu
[3] School of Computing and Communications, Lancaster University
h.rahmani@lancaster.ac.uk
[4] ByteDance
anranwang1991@gmail.com
[5] School of Computer Science and Technology, Harbin Institute of Technology
wangjinghua@hit.edu.cn

Abstract. The goal of fine-grained action recognition is to successfully discriminate between action categories with subtle differences. To tackle this, we derive inspiration from the human visual system which contains specialized regions in the brain that are dedicated towards handling specific tasks. We design a novel Dynamic Spatio-Temporal Specialization (DSTS) module, which consists of specialized neurons that are only activated for a subset of samples that are highly similar. During training, the loss forces the specialized neurons to learn discriminative fine-grained differences to distinguish between these similar samples, improving fine-grained recognition. Moreover, a spatio-temporal specialization method further optimizes the architectures of the specialized neurons to capture either more spatial or temporal fine-grained information, to better tackle the large range of spatio-temporal variations in the videos. Lastly, we design an Upstream-Downstream Learning algorithm to optimize our model's dynamic decisions during training, improving the performance of our DSTS module. We obtain state-of-the-art performance on two widely-used fine-grained action recognition datasets.

Keywords: Action recognition, fine-grained, dynamic neural networks.

1 Introduction

Fine-grained action recognition involves distinguishing between similar actions with only subtle differences, e.g., "cutting an apple in a kitchen" and "cutting a pear in a kitchen". This is significantly more challenging than coarse-grained

⋆ equal contribution
⋆⋆ corresponding author

classification, where the action classes can be "cutting something in a kitchen" and "playing in a gym". The higher inter-class similarity in the fine-grained setting makes it a challenging task, which coarse-grained backbones and methods struggle to overcome.

To tackle the challenging fine-grained action recognition task, we derive inspiration from the remarkable human visual system which has good fine-grained recognition capabilities. Importantly, our visual system comprises of specialized neurons that are activated only under some specific circumstances, as shown by previous works [33, 23]. For example, for enhanced recognition of humans which is crucial for social behaviour, human brains have developed a set of cortical regions specialized for processing faces [33, 23]. These specialized regions fire only when our attention focuses on human faces, while specific sub-regions are further specialized to fire specifically for processing face parts [25], eye gazes and expressions [12], and identity [27, 11].

Inspired by the specialization of neurons in the human brain, we improve fine-grained recognition capabilities of a neural network by using specialized parameters that are only activated on a subset of the data. More specifically, we design a novel Dynamic Spatio-Temporal Specialization (DSTS) module which consists of *specialized neurons* that are only activated when the input is within their area of specialization (as determined by their individual *scoring kernels*). In particular, a synapse mechanism dynamically activates each specialized neuron only on a subset of samples that are highly similar, such that only fine-grained differences exist between them. During training, in order to distinguish among that particular subset of similar samples, the loss will push the specialized neurons to focus on exploiting the fine-grained differences between them. We note that previous works on fine-grained action recognition [48, 38, 52] have not explicitly considered such specialization of parameters. These works [48, 38, 52] propose deep networks where all parameters are generally updated using all data samples, and thus, during training, the loss tends to encourage those models to pick up more common discriminative cues that apply to the more common samples, as opposed to various fine-grained cues that might be crucial to different subsets of the data.

Fig. 1. Key frames of samples taken from the Something-Something-v2 dataset [10]. (Top) Fine-grained differences lie more in the spatial aspects of the two actions, as shown in the green box. To distinguish between these two actions, we need to focus on whether the water in the cup overflows in the final key frame, which can be quite subtle. (Bottom) Fine-grained differences lie mainly in the temporal aspects of the two actions, where we need to focus on the movement (denoted with yellow arrows) of the object across all key frames. Best viewed in colour.

Another interesting insight comes from the human primary visual cortex, where there are neurons that are observed to be specialized in temporal or spatial aspects [43, 24]. Magnocellular, or M cells, are observed to be specialized to detect movement, e.g., speed and direction. Parvocellular, or P cells, are important for spatial resolution, e.g., shape, size and color. Together, they effectively allow humans to distinguish between actions.

Spatial and temporal specialization has clear benefits for fine-grained action recognition. As observed in Fig 1, some fine-grained differences lie mainly in the temporal aspects of two actions, e.g., "Pulling something from right to left" and "Pulling something from left to right". In this case, a greater emphasis on the temporal dimension of each video will lead to better recognition performance. In contrast, some fine-grained differences lie more in the spatial aspects of two actions, e.g., "Pouring something into something" and "Pouring something into something and it overflows". In this case, greater emphasis on the spatial dimension can improve the performance.

To allow our module to efficiently and effectively handle fine-grained differences over a large range of spatio-temporal variations, we design a *spatio-temporal specialization* method that additionally provides specialized neurons with spatial or temporal specializations. To achieve such specialization, we explicitly design the specialized neurons to focus only on one single aspect (spatial or temporal) for each channel of the input feature map at a time, forcing the neurons to exploit fine-grained differences between similar samples in that specific aspect, leading to higher sensitivity towards these fine-grained differences. Specifically, this is implemented using *gates* that determine whether a *spatial operator* or a *temporal operator* is used to process each input channel. By adjusting their gate parameters, specialized neurons that benefit from discerning spatial or temporal patterns adapt their architectures to use the corresponding operator across more channels. Eventually, the set of specialized neurons will have diversified architectures and specializations focusing on different spatial and temporal aspects, which collectively are capable of handling a large variety of spatial and temporal fine-grained differences.

During end-to-end training of our module, we jointly train two types of parameters: *upstream parameters* (i.e., scoring kernels and gate parameters) that make dynamic decisions and *downstream parameters* (i.e., spatial and temporal operators) that process input, which can be challenging as *upstream parameters themselves also affect the training of downstream ones*. Hence, we design an Upstream-Downstream Learning (UDL) algorithm to optimize upstream parameters to learn how to make decisions that positively affect the training of downstream parameters, improving the performance of our DSTS module.

2 Related Work

2.1 Action Recognition

Action recognition involves taking an action video clip as input and predicting the class of the action. Many methods have been proposed to tackle this task,

including the two-stream [30], TSN [35], TRN [51], TSM [19], TPN [44], LTC [34], I3D [3], S3D [42], SlowFast [7], X3D [6], NL [36], GST [22], Tx [9], TimeSformer [2], ViViT [1], MViT-B [5], and Swin Transformer [20].

2.2 Fine-grained Action Recognition

In comparison, fine-grained action recognition, where actions have lower inter-class differences, has been relatively less explored. Datasets such as Something-Something-v2 [10] and Diving48 [18] have been curated for this purpose.

Interaction Part Mining [52] mines mid-level parts, connects them to form a large spatio-temporal graph and mines interactions within the graph. LFB [38] employs a long-term feature bank for detailed processing of long videos that provides video-level contextual information at every time step. FineGym [28] has found that coarse-grained backbones lack the capability to capture complex temporal dynamics and subtle spatial semantics for their fine-grained dataset. TQN [48] casts fine-grained action recognition as a query-response task, where the model learns query vectors that are decoded into response vectors by a Transformer.

Different from previous works that do not explicitly consider specialized parameters, we propose a novel dynamic DSTS module that trains and selects specialized neurons for fine-grained action recognition. Furthermore, we investigate a novel spatio-temporal specialization scheme that optimizes architectures of the specialized neurons to focus more on spatial or temporal aspects, further specializing them for improved fine-grained action recognition.

2.3 Dynamic Neural Networks

Dynamic neural networks generally adapt their parameters or structures according to the input. Typical approaches include generating weights with a subnetwork, dynamically selecting network depth and dynamically selecting network widths [46, 37, 13, 40, 50]. On videos, several methods [41, 39] adaptively select video frames for the sake of efficiency. GSM [31] learns to adaptively route features from a 2D-CNN through time and combine them. TANet [21] employs a dynamic video aggregation kernel that adds global video information to 2D convolutions.

Different from these methods, our DSTS module focuses on improving performance on fine-grained action recognition. We design a novel synapse mechanism that activates each specialized neuron only on samples that are highly similar, pushing them to pick up relevant fine-grained differences to distinguish between these similar samples. We further propose spatio-temporal specialization of our specialized neurons, which to the best of our knowledge, has not yet been explored in previous works.

2.4 Kernel Factorization

Kernel Factorization generally involves factorizing a 3D spatio-temporal convolution into a 2D spatial convolution plus a 1D temporal convolution, such

as in P3D [26], S3D [42] and R(2+1)D [32]. In GST [22], 3D convolutions are decomposed into a fixed combination of parallel spatial and temporal convolutions. In these works, the kernel factorization leads to improved effectiveness and efficiency.

Here, we propose a novel DSTS module that dynamically activates the most relevant specialized neuron. Different from previous works, our specialized neurons learn to select a spatial or temporal operator for each channel, to better handle the corresponding fine-grained differences between similar samples, for fine-grained action recognition.

3 Proposed Method

3.1 Overview

In fine-grained action recognition, actions from different classes can be highly similar, with only fine-grained differences between them. Such fine-grained differences might not be effectively learnt by parameters that are trained on all samples, as they will tend to capture common discriminative cues that occur more commonly throughout the data, instead of various fine-grained cues, each of which might only be relevant in a small subset of the data [15]. Thus, to improve performance on fine-grained action recognition, we propose to employ specialized parameters in our model. These specialized parameters are pushed to gain specialized capabilities in identifying fine-grained differences by being trained only on a subset of the data that contains highly similar samples.

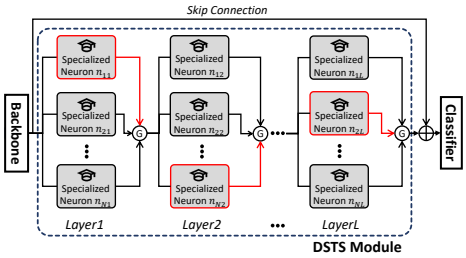

Fig. 2. Illustration of the proposed DSTS module, which processes features extracted from a backbone. There are L layers within the DSTS module, each comprising N specialized neurons (grey rectangles). When a feature map X is fed into the j-th DSTS layer, impulse values v_{ij} from each specialized neuron n_{ij} are first calculated, and the specialized neuron with the highest impulse value in that layer is activated (indicated with red arrows) using the Gumbel-Softmax technique (indicated with Ⓖ). A skip connection adds general features from the backbone to the output of the DSTS module (indicated with $\bigoplus$), before being fed into the classifier.

Our DSTS module achieves this specialization through the dynamic usage of blocks of parameters called *specialized neurons*, which can be observed in Fig. 2. For each input sample, only one specialized neuron (i.e., the neuron with the most relevant specialization) in each layer is activated to process the sample – this dynamic activation occurs in what we call the *synapse mechanism*. Crucially, we design the synapse mechanism such that each specialized neuron is only activated on a subset of samples that are similar, with only fine-grained differences between them. During training, since each specialized neuron is only trained on a subset of the data that contains similar samples, the training loss

will push the specialized neuron to learn to handle the fine-grained information relevant to these samples, instead of learning more common discriminative cues that are applicable to the more common samples. Hence, each specialized neuron gains specialized capability that is highly effective at classifying a particular subset of samples, leading to improved fine-grained recognition performance.

Moreover, considering that fine-grained differences between similar samples might exist in more spatial or temporal aspects, we further propose spatio-temporal specialization in the specialized neurons, to further optimize their architectures. By explicitly forcing the specialized neurons to focus on spatial or temporal aspects for each channel of the input feature map, they are pushed to exploit fine-grained differences in that specific aspect, leading to better sensitivity towards the fine-grained differences in that aspect. Such channel-wise decisions on spatial or temporal specializations are learned in an end-to-end manner for improved performance. Lastly, we further improve the generalization capability of our DSTS module by proposing *Upstream-Downstream Learning*, where the model parameters involved in making its dynamic decisions are meta-learned.

Next, we formally introduce the DSTS module which is illustrated in Fig. 2. Setting batch size to 1 for simplicity, we assume that the pre-trained backbone outputs a feature map $X \in \mathbb{R}^{N_{in} \times N_t \times N_h \times N_w}$, where N_{in}, N_t, N_h, N_w represent the channel, temporal, height and width dimensions of the feature map, respectively. The DSTS module consists of L layers, with each layer comprising of N specialized neurons. We define the i-th specialized neuron in the j-th layer as n_{ij}, which is shown in detail in Fig. 3. Each specialized neuron n_{ij} has a scoring kernel $m_{ij} \in \mathbb{R}^{N_{out} \times N_{in} \times 1 \times 1 \times 1}$ (with the size of $1 \times 1 \times 1$ for efficiently encoding information from all channels of feature map X), a spatial operator consisting of a convolutional kernel $S_{ij} \in \mathbb{R}^{N_{out} \times N_{in} \times 1 \times 3 \times 3}$ (2D on the spatial domain), a temporal operator consisting of a convolutional kernel $T_{ij} \in \mathbb{R}^{N_{out} \times N_{in} \times 3 \times 1 \times 1}$ (1D on the temporal domain) and gates $g_{ij} \in \mathbb{R}^{N_{in}}$.

3.2 DSTS Layer

In this subsection, we describe a single DSTS layer. For clarity, we describe the first DSTS layer and omit the layer index, using n_i to represent the i-th specialized neuron (which consists of m_i, S_i, T_i and g_i) in this DSTS layer.

Synapse Mechanism The synapse mechanism is the crucial step that dynamically activates the specialized neuron with the most relevant specialization for the given input feature map X. Importantly, similar feature maps should activate the same specialized neurons, so that each specialized neuron is pushed to specialize in fine-grained differences to distinguish between these similar feature maps during training.

To implement the synapse mechanism to achieve the above-mentioned specialization effect, we include a *scoring kernel* m_i in each specialized neuron n_i that is applied on the input feature map X in a step that we call the *scoring*

convolution. The resulting output is summed to produce a relevance score (which we call an *impulse* v_i) between the input feature map X and the fine-grained specialization capabilities of the specialized neuron n_i. The higher the impulse produced by a specialized neuron, the higher the relevance of the specialized neuron's knowledge to the input feature, and the more likely it will be activated.

In the first step, to calculate the relevance scores between a specialized neuron n_i and a feature map X, we first apply a scoring convolution using the scoring kernel m_i on X:

$$q_i = m_i(X), \tag{1}$$

where we slightly abuse the notation to let $m_i(\cdot)$ denote the scoring convolution function applied to an input using scoring kernel m_i (we adopt this notation for all convolution functions in this work) and $q_i \in \mathbb{R}^{N_{out} \times Q_t \times Q_h \times Q_w}$ is an intermediate representation with Q_t, Q_h, Q_w being the resulting temporal, width and height dimensions.

We then sum all elements in q_i to get the impulse v_i of specialized neuron n_i.

$$v_i = \sum_{u_c=1}^{N_{out}} \sum_{u_t=1}^{Q_t} \sum_{u_h=1}^{Q_h} \sum_{u_w=1}^{Q_w} q_{i,u_c,u_t,u_h,u_w} \tag{2}$$

We conduct the above process (Eq. 1 and Eq. 2) for all scoring kernels $\{m_i\}_{i=1}^{N}$ of the N specialized neurons in the DSTS layer to obtain the complete set of impulse values $\mathcal{V}$:

$$\mathcal{V} = \{v_i\}_{i=1}^{N}. \tag{3}$$

Finally, we apply the Gumbel-Softmax technique [14] on $\mathcal{V}$ to select a specialized neuron to activate. The selection to activate specialized neuron n_a is made by producing a one-hot vector with a 1 at the selected index a. During training, the Gumbel-Softmax allows gradients to backpropagate through this selection mechanism. During testing, the activated specialized neuron n_a is the one with the highest impulse within $\mathcal{V}$, and has the most relevant specialization to discriminate between samples similar to the input X.

We remark that this synapse mechanism is crucial for the specialization of the specialized neurons. As convolutional filters tend to produce similar responses for similar feature maps [47, 45], q_i and v_i tend to be similar for similar feature maps. Hence, during training, similar feature maps are highly likely to produce high impulse scores for (and activate) the same specialized neuron; this neuron will thus be updated using only a subset of similar samples, which pushes this neuron to specialize in fine-grained differences to distinguish between them.

Spatio-Temporal Specialization Intuitively, after n_a is activated, we can simply apply a 3D convolution kernel (corresponding to n_a) on X to extract the spatio-temporal information. Yet, in fine-grained action recognition, the fine-grained differences between actions can exist in more spatial or temporal aspects of actions, which require emphasis along their respective dimensions for effective

discrimination. Motivated by this, instead of optimizing the parameters within a 3D kernel architecture, we additionally optimize the architectures of the specialized neurons to specialize in focusing on either more spatial or more temporal fine-grained information.

More concretely, our spatio-temporal specialization method adapts the architectures of the specialized neurons to utilize either a *spatial operator* or a *temporal operator* for each input channel. The spatial operator uses a 2D convolution that focuses on the spatial aspects of the feature map while the temporal operator uses a 1D convolution that focuses on the temporal aspects. To achieve spatial or temporal specialization, we explicitly restrict the specialized neurons to choose between spatial or temporal operators for each input channel. During training, this design forces each specialized neuron to exploit fine-grained differences in each channel between similar samples in the chosen aspect, leading to better sensitivity towards these fine-grained differences. Since different channels of the input feature map can convey different information, which might lie in the spatial or temporal aspects, we let our model adapt its architecture to select the operator in each channel that would lead to greater discriminative capability. Such architectural decisions (spatial or temporal) for each channel are learned by the *gate* parameters. When it is beneficial for the specialized neuron to focus more on a certain fine-grained aspect, the gates will learn to use the corresponding operator across more channels, pushing for higher sensitivity towards that aspect for improved discriminative capability. The efficacy of this channel-wise design for spatio-temporal specialization is investigated empirically along with other baselines in Section 4.3.

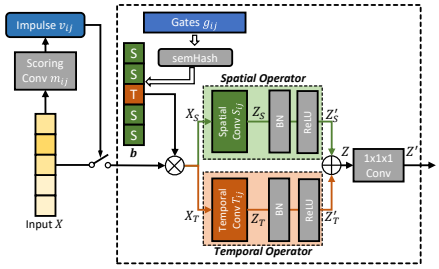

Fig. 3. Illustration of a specialized neuron n_{ij}. A scoring convolution using m_{ij}, followed by a summation, produces impulse v_{ij} that is used to determine if n_{ij} is activated. Gate parameters g_{ij} are used to generate $\mathbf{b}$ using the Improved Semhash method, which determines (using channel-wise multiplication $\bigotimes$) if each input channel uses the spatial operator's kernel S_{ij} (in green) or the temporal operator's kernel T_{ij} (in orange). After the processing of the spatial and temporal operators, both features Z'_S and Z'_T are added (indicated with $\bigoplus$) and fused using a $1 \times 1 \times 1$ convolution to get output Z'.

Spatio-temporal Architectural Decisions using Gates. This step takes place after the synapse mechanism, where a specialized neuron n_a is activated. The specialized neuron's gate parameters g_a consists of N_{in} elements, with each element corresponding to one input channel. Each gate parameter determines if the corresponding channel is processed using the spatial or temporal operator.

During the forward pass, we sample binary decisions from the gate parameters g_a using the Improved Semhash method [16, 17, 4], obtaining a binary vector $\mathbf{b} \in \{0, 1\}^{N_{in}}$. Improved Semhash allows us to train gate parameters g_a in an

end-to-end manner. We opt for the Improved Semhash instead of the Gumbel-Softmax here as we can use less parameters (N_{in} instead of $2N_{in}$). We denote the l-th element of $\mathbf{b}$ as b_l. If $b_l = 0$, then the corresponding input channel l will use the spatial operator. While if $b_l = 1$, then the corresponding input channel l will use the temporal operator. More details of Improved Semhash can be found in the Supplementary.

Specialized Spatio-Temporal Processing. After obtaining channel-wise architectural decisions $\mathbf{b}$, we can commence with the channel-wise selection of input feature map X to obtain features X_S and X_T as follows, which will be used for learning fine-grained spatial and temporal information respectively:

$$X_S = (\mathbf{1} - \mathbf{b}) \cdot X, \tag{4}$$

$$X_T = \mathbf{b} \cdot X, \tag{5}$$

where $\mathbf{1}$ is a vector of 1's of size N_{in}, and $\cdot$ refers to multiplication along the channel dimension while treating each element of $\mathbf{b}$ and ($\mathbf{1}$ - $\mathbf{b}$) as a channel. Using X_S and X_T, spatial and temporal outputs Z_S, Z_T are obtained using the respective spatial and temporal kernels S_a, T_a within n_a:

$$Z_S = S_a(X_S), \tag{6}$$

$$Z_T = T_a(X_T), \tag{7}$$

where Z_S denotes features that capture spatial information of input feature map X, while Z_T denotes features that capture temporal information. Z_S and Z_T are then fed to a batch normalization and a ReLU activation layer. We denote the two output features as Z'_S and Z'_T. The output feature map Z is obtained by adding Z'_S and Z'_T:

$$Z = Z'_S + Z'_T \tag{8}$$

Lastly, a $1 \times 1 \times 1$ convolution is applied to Z to fuse both spatial and temporal features. These fused features Z' are then fed to the next DSTS layer or classifier.

Spatio-temporal specialization allows specialized neurons to focus on either more spatial or temporal fine-grained information. If a specialized neuron n_i is activated on a subset of similar samples with fine-grained spatial differences, encoding more spatial information by applying the spatial operator on more channels will tend to be more effective, and g_i will be trained to produce more 0's in $\mathbf{b}$. On the other hand, if the samples in the subset contain more fine-grained temporal differences, the model will learn to apply the temporal operator across more channels, by optimizing g_i to produce more 1's in $\mathbf{b}$ for better fine-grained action recognition. It is also possible that the spatial and temporal aspects are equally important to discriminate similar actions. In this case, g_i will be optimized to handle both spatial and temporal fine-grained information.

3.3 Upstream-Downstream Learning

To further improve the performance of our DSTS module, we design a UDL algorithm that better optimizes the model parameters involved in making dynamic decisions, which we call *upstream parameters*. These upstream parameters

(i.e., scoring kernels m and gate parameters g) that make dynamic decisions and *downstream parameters* (i.e., spatial and temporal operators S and T) that process input, are jointly trained during our end-to-end training, which can be challenging as *upstream parameters themselves also affect the training of downstream ones*. This is because, upstream parameters determine which downstream parameters will be used, and consequently updated. Hence, we use meta-learning [8, 29] to optimize upstream parameters while taking their downstream effects into account, leading to the improved learning of downstream parameters and overall improved performance.

There are three steps in our meta-learning algorithm. In the first step, we simulate an update step by updating downstream parameters while keeping upstream parameters frozen. This simulates the training process of the downstream parameters when the current set of upstream parameters are used to make dynamic decisions. In the crucial second step, we evaluate the model's performance on held-out samples in a validation set, which estimates model performance on unseen samples. The second-order gradients (with respect to upstream parameters) from this evaluation provide feedback on how upstream parameters can be updated such that their *dynamic decisions during training can improve the learning process of downstream parameters*, leading to better performance on unseen samples. In the final step, downstream parameters are optimized using the *meta-optimized upstream parameters*, which now make dynamic decisions in the model such that downstream parameters are able to benefit more from training and have improved (testing) performance.

More concretely, in each iteration, we sample two mini-batches from the training data: training samples D_{train} and validation samples D_{val}. The two mini-batches should not contain overlapping samples, as we want to use D_{val} to estimate performance on unseen samples. The algorithm proceeds in three steps:

Firstly, a **Simulated Update Step** updates downstream parameters d using supervised loss ℓ on D_{train}.

$$\hat{d} = d - \alpha \nabla_d \ell(u, d; D_{train}), \tag{9}$$

where α is a learning rate hyperparameter, while u and d denote the upstream and downstream parameters respectively. We keep upstream parameters u fixed in this step.

Secondly, a **Meta-Update Step** evaluates the updated model on D_{val}. We update upstream parameters u using the second-order gradients with respect to u when they were used to make decisions in the first Simulated Update Step, as follows:

$$u' = u - \alpha \nabla_u \ell(\hat{u}, \hat{d}; D_{val}), \tag{10}$$

where $\hat{u}$ is a copy of u, but no gradients are computed with respect to $\hat{u}$. We denote it this way, because the same set of u parameters are used twice (in Eq. 9 and Eq. 10), and we want to compute second-order gradients ∇_u with respect to u in Eq. 9, not first-order gradients with respect to $\hat{u}$ in Eq. 10. These second-order gradients ∇_u provide feedback on how to adjust u such that their dynamic decisions lead to better training of the downstream parameters (as simulated

in the Simulated Update Step), resulting in improved performance on unseen samples. d is not updated in this step.

Finally, d is updated in the **Actual Update Step** while keeping u' frozen.

$$d' = d - \alpha \nabla_d \ell(u', d; D_{train}) \tag{11}$$

One iteration of this algorithm concludes here, and we obtain updated parameters u' and d'. An outline of the algorithm is shown in the Supplementary.

4 Experiments

We conduct experiments using our proposed DSTS module on two popular fine-grained action recognition datasets, i.e., the Something-Something v2 dataset (SSV2) [10] and Diving48 dataset [18].

SSV2 [10] is a large dataset, containing approximately 220k videos across 174 different classes. It consists of crowd-sourced clips that show humans performing basic actions with various types of everyday objects. The difference between classes could lie in fine-grained spatial or temporal details, as depicted in Fig. 1. Following [10, 20, 49], we split the data into 169k training and 27k test videos.

Diving48 [18] contains approximately 18k trimmed video clips of 48 classes of competitive diving sequences. There are fine-grained differences between the 48 classes, which could exist at takeoff, in flight, at entry, or a combination of them in the diving sequences, making it a challenging classification task. Following [18, 48], we split the data into 16k training and 2k test videos. Following [48], we use the cleaned (v2) labels released in Oct 2020.

4.1 Implementation details

To evaluate the efficacy of the proposed DSTS module, Swin-B transformer [20] and TPN [44] are used as the backbone networks. In our experiments, each DSTS layer contains 10 specialized neurons ($N = 10$) and the DSTS module has 3 layers ($L = 3$). The dimensions of the input X, such as N_{in}, N_t, N_h, N_w are determined by different backbone networks, and we set $N_{out} = N_{in}$. Thus, the shape of S_{ij}, T_{ij}, g_{ij} and m_{ij} for each n_{ij} are dependent on the backbones.

The experiments are conducted on 8 Nvidia V100 GPUs with batch size $B = 8$. We follow the experimental settings of Video Swin Transformer [20], using the AdamW optimizer and setting the initial learning rate α as 3×10^{-4}. For TPN, we follow the experimental settings in [44], using the SGD optimizer and setting the initial learning rate α as 0.01. We compute cross-entropy loss as the supervised loss ℓ for classification.

During **training**, using the Gumbel-Softmax and Improved Semhash techniques for selection of specialized neurons and operators, our model is end-to-end trainable. We set Gumbel-Softmax temperature $\tau = 1$, and the noise applied to Improved SemHash is sampled from a standard Gaussian distribution.

During **testing**, given an input feature map X, impulse values $\{v_{ij}\}_{i=1}^{N}$ are computed for all N specialized neurons in each layer j. However, because we

Table 1. Top-1 and Top-5 scores (%) on SSv2. Type "C" indicates CNN-based architectures and "T" indicates Transformer-based architectures. Our DSTS module improves Top-1 accuracy of TPN by 2.5% and Swin-B by 2.2%.

Method	Type	Top-1	Top-5
SlowFast [7]	C	63.1	87.6
TPN [44]	C	64.7	88.1
ViViT-L [1]	T	65.4	89.8
TSM (Two-stream) [19]	C	66.6	91.3
MViT-B [5]	T	67.7	90.9
Swin-B [20]	T	69.6	92.7
TPN w/ DSTS	C	67.2	89.2
Swin-B w/ DSTS	T	**71.8**	**93.7**

do not require gradients this time, the input X is only processed by the best-matching specialized neuron n_{aj} of each layer j, to obtain output Z'_{aj}. Notably, no noise is added to Gumbel-Softmax and Improved Semhash during inference.

4.2 Experiment Results

Results on SSv2. Following [20, 1, 19], we report Top-1 and Top-5 accuracy scores across all models on the test set of SSv2. Results are shown in Table 1. As both CNNs and Transformers are used to tackle action recognition, we test DSTS on a CNN-based architecture (TPN [44]) and a Transformer-based architecture (Swin-B [20]) to investigate if our DSTS module provides performance gains on both types of architectures.

Adding our DSTS module to baseline architectures leads to improved performance on both architectures. Adding DSTS to TPN (**TPN w/ DSTS**), the performance of TPN improves by 2.5%, achieving a Top-1 accuracy of 67.2%. To the best of our knowledge, this performance is state-of-the-art among CNN-based architectures, surpassing even the performance of the two-stream TSM which utilizes additional optical flow information. This shows that DSTS can improve performance for CNN-based backbones on fine-grained action recognition. Adding DSTS to Swin-B (**Swin-B w/ DSTS**) improves Top-1 accuracy by 2.2%, achieving a new state-of-the-art of 71.8%, showing that DSTS can help improve fine-grained action recognition on Transformer-based backbones as well. Qualitative results and visualizations have been placed in the Supplementary.

Results on Diving48. Following [48], we report Top-1 accuracy and mean accuracy per class across all models on Diving48 dataset. Results are shown in Table 2. Using DSTS module leads to significant improvements on Diving48 as well. It achieves a Top-1 improvement of 2.2% on TPN and 2.5% on Swin-B. TPN w/ DSTS achieves state-of-the-art result of 88.4% Top-1 accuracy.

4.3 Ablation Studies

We conduct extensive ablation studies to evaluate the importance of certain design choices. Ablation studies are conducted on Diving48, using TPN as a backbone. More experiments are placed in the Supplementary.

Table 2. Top-1 and Class-wise accuracy scores (%) on Diving48. Our DSTS module improves Top-1 accuracy of TPN by 2.2% and Swin-B by 2.5%.

Method	Type	Top-1	Class-wise Acc
I3D [3]	C	48.3	33.2
TSM (Two-stream) [19]	C	52.5	32.7
GST [22]	C	78.9	69.5
TQN [48]	T	81.8	74.5
Swin-B [20]	T	80.5	69.7
TPN [44]	C	86.2	76.0
Swin-B w/ DSTS	T	83.0	71.5
TPN w/ DSTS	C	**88.4**	**78.2**

1) Spatio-temporal specialization. We evaluate the impact of spatio-temporal specialization on our DSTS module, and the results are shown in Table 3. It can be observed that DSTS module with spatio-temporal specialization (**DSTS w/ STS**) performs better than DSTS without it (**DSTS w/o STS**), showing its effectiveness. For DSTS w/o STS, only one operator, i.e., a 3D convolution with batch normalization and ReLU, within each n_{ij} is employed to process X. Besides, when we remove gates g_{ij} (**DSTS w/o Gates**), and let all specialized neurons have the same factorized architecture (the channels are split into two fixed halves, to which the spatial and temporal operators are applied respectively), the performance decreases by 1.1%. This shows that our gates learn channel-wise architectures that are more specialized and effective for fine-grained recognition, compared to fixed architectures.

Table 3. Evaluation results (%) on the impact of spatio-temporal specialization of DSTS modules on Diving48.

Method	Top-1	Class-wise Acc
DSTS w/o STS	87.2	76.5
DSTS w/o Gates	87.3	76.7
DSTS w/ STS	88.4	78.2

2) Synapse Mechanism. We investigate the impact of the synapse mechanism, and results are shown in Table 4. Following our method with the dynamic synapse mechanism and activating the most relevant specialized neuron at each layer (**w/ Synapse Mechanism**) achieves better results compared to activating all specialized neurons and averaging their outputs (**w/o Synapse Mechanism**), which is a non-dynamic design with the same number of parameters as our method. This shows that the performance improvement comes from our synapse mechanism and its dynamic design, and not the additional parameters. This improvement is because, unlike w/o Synapse Mechanism which trains all neurons on all data samples and tends to learn more common discriminative cues that apply to the more common samples, our DSTS module trains each specialized neuron only on a subset of similar samples, explicitly pushing them to gain better specialized fine-grained abilities.

Table 4. Evaluation results (%) on the impact of the synapse mechanism on Diving48.

Method	Top-1	Class-wise Acc	Model Size
Baseline TPN	86.2	76.0	63M
w/o Synapse Mechanism	86.5	76.4	75M
w/ Synapse Mechanism	88.4	78.2	75M

3) Upstream-Downstream Learning. We conduct experiments to evaluate the performance gains from our UDL method, and the results can be seen in Table 5.

We observe that our UDL method (**DSTS w/ UDL**) improves performance over using backpropagation in a single step (**DSTS w/o UDL**). We emphasize that such these performance gains are achieved using only slightly more training time, which is reported in the Supplementary.

Table 5. Evaluation results (%) on the impact of the UDL method on Diving48.

Method	Top-1	Class-wise Acc
DSTS w/o UDL	87.4	76.7
DSTS w/ UDL	88.4	78.2

4) Number of specialized neurons in each DSTS layer. We evaluate the impact of using different numbers of specialized neurons in each DSTS layer, and the results are shown in Table 6. When N is low (e.g., $N = 5$), using more specialized neurons in each DSTS layer (e.g., $N = 10$) improves the performance, which can be explained by the increase in representational capacity. More precisely, when there are more specialized neurons, each one can afford to be more specialized towards a smaller subset of data, which improves their capability. We use $N = 10$ as this improvement effect tapers off when N is increased beyond 10.

Table 6. Evaluation results (%) for different numbers of specialized neurons N in each DSTS module on Diving48.

N	Top-1	Class-wise Acc
5	87.3	76.2
10	88.4	78.2
15	88.3	78.2

5) Number of DSTS layers. We also evaluate the impact of varying L, i.e., stacking different numbers of DSTS layers, and results are shown in Table 7. As expected, stacking more DSTS layers leads to better performance. This is because, the DSTS module with more layers could have greater representational capacity to process more complex and fine-grained cues. When we increase L from 1 to 3, we obtain an improvement of 0.9%, and increasing L further does not lead to further improvement. We thus set $L = 3$.

Table 7. Evaluation results (%) for different numbers of DSTS layers L on Diving48.

L	Top-1	Class-wise Acc
1	87.5	76.8
3	88.4	78.2
5	88.2	78.2

5 Conclusion

In this paper, we have proposed a novel DSTS module consisting of dynamically activated specialized neurons for fine-grained action recognition. Our spatio-temporal specialization method optimizes the architectures of specialized neurons to focus more on spatial or temporal aspects. Our UDL procedure further improves the performance of our DSTS module. We obtain state-of-the-art fine-grained action recognition performance on two popular datasets by adding DSTS modules to baseline architectures.

Acknowledgement This work is supported by National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-100E-2020-065), Ministry of Education Tier 1 Grant and SUTD Startup Research Grant. This work is also partially supported by Natural Science Foundation of China (NSFC) under the Grant no. 62172285. The research is also supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lucic, M., Schmid, C.: Vivit: A video vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6836–6846 (October 2021) 4, 12
2. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? (2021) 4
3. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6299–6308 (2017) 4, 13
4. Chen, Z., Li, Y., Bengio, S., Si, S.: You look twice: Gaternet for dynamic filter selection in cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9172–9180 (2019) 8
5. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. arXiv preprint arXiv:2104.11227 (2021) 4, 12
6. Feichtenhofer, C.: X3d: Expanding architectures for efficient video recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 203–213 (2020) 4
7. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6202–6211 (2019) 4, 12
8. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017) 10
9. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 244–253 (2019) 4
10. Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al.: The" something something" video database for learning and evaluating visual common sense. In: Proceedings of the IEEE international conference on computer vision. pp. 5842–5850 (2017) 2, 4, 11
11. Haxby, J.V., Hoffman, E.A., Gobbini, M.I.: The distributed human neural system for face perception. Trends in cognitive sciences **4**(6), 223–233 (2000) 2
12. Hoffman, E.A., Haxby, J.V.: Distinct representations of eye gaze and identity in the distributed human neural system for face perception. Nature neuroscience **3**(1), 80–84 (2000) 2
13. Hua, W., Zhou, Y., De Sa, C., Zhang, Z., Suh, G.E.: Channel gating neural networks (2019) 4
14. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016) 7
15. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. Journal of Big Data **6**(1), 1–54 (2019) 5
16. Kaiser, Ł., Bengio, S.: Discrete autoencoders for sequence models. arXiv preprint arXiv:1801.09797 (2018) 8
17. Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., Shazeer, N.: Fast decoding in sequence models using discrete latent variables. In: International Conference on Machine Learning. pp. 2390–2399. PMLR (2018) 8
18. Li, Y., Li, Y., Vasconcelos, N.: Resound: Towards action recognition without representation bias. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 513–528 (2018) 4, 11

19. Lin, J., Gan, C., Han, S.: Tsm: Temporal shift module for efficient video understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7083–7093 (2019) 4, 12, 13
20. Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H.: Video swin transformer. arXiv preprint arXiv:2106.13230 (2021) 4, 11, 12, 13
21. Liu, Z., Wang, L., Wu, W., Qian, C., Lu, T.: Tam: Temporal adaptive module for video recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13708–13718 (2021) 4
22. Luo, C., Yuille, A.L.: Grouped spatial-temporal aggregation for efficient action recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5512–5521 (2019) 4, 5, 13
23. Minxha, J., Mosher, C., Morrow, J.K., Mamelak, A.N., Adolphs, R., Gothard, K.M., Rutishauser, U.: Fixations gate species-specific responses to free viewing of faces in the human and macaque amygdala. Cell Reports **18**(4), 878–891 (2017). https://doi.org/https://doi.org/10.1016/j.celrep.2016.12.083, https://www.sciencedirect.com/science/article/pii/S2211124716318058 2
24. Nolte, J., Vanderah, T., Gould, D.: Nolte's the human brain: An introduction to its functional anatomy (2016) 3
25. Pitcher, D., Walsh, V., Yovel, G., Duchaine, B.: Tms evidence for the involvement of the right occipital face area in early face processing. Current Biology **17**(18), 1568–1573 (2007) 2
26. Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: proceedings of the IEEE International Conference on Computer Vision. pp. 5533–5541 (2017) 5
27. Rotshtein, P., Henson, R.N., Treves, A., Driver, J., Dolan, R.J.: Morphing marilyn into maggie dissociates physical and identity face representations in the brain. Nature neuroscience **8**(1), 107–113 (2005) 2
28. Shao, D., Zhao, Y., Dai, B., Lin, D.: Finegym: A hierarchical video dataset for fine-grained action understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2616–2625 (2020) 4
29. Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D.: Meta-weight-net: Learning an explicit mapping for sample weighting (2019) 10
30. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1. p. 568–576. NIPS'14, MIT Press, Cambridge, MA, USA (2014) 4
31. Sudhakaran, S., Escalera, S., Lanz, O.: Gate-shift networks for video action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1102–1111 (2020) 4
32. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 6450–6459 (2018) 5
33. Tsao, D.Y., Moeller, S., Freiwald, W.A.: Comparing face patch systems in macaques and humans. Proceedings of the National Academy of Sciences **105**(49), 19514–19519 (2008) 2
34. Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. IEEE transactions on pattern analysis and machine intelligence **40**(6), 1510–1517 (2017) 4
35. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: European conference on computer vision. pp. 20–36. Springer (2016) 4

36. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7794–7803 (2018) 4
37. Wang, X., Yu, F., Dou, Z.Y., Darrell, T., Gonzalez, J.E.: Skipnet: Learning dynamic routing in convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 409–424 (2018) 4
38. Wu, C.Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., Girshick, R.: Long-term feature banks for detailed video understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 284–293 (2019) 2, 4
39. Wu, Z., Li, H., Xiong, C., Jiang, Y.G., Davis, L.S.: A dynamic frame selection framework for fast video recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 4
40. Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R.: Blockdrop: Dynamic inference paths in residual networks. In: CVPR. pp. 8817–8826 (2018) 4
41. Wu, Z., Xiong, C., Ma, C.Y., Socher, R., Davis, L.S.: Adaframe: Adaptive frame selection for fast video recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1278–1287 (2019) 4
42. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Proceedings of the European conference on computer vision (ECCV). pp. 305–321 (2018) 4, 5
43. Xu, X., Ichida, J.M., Allison, J.D., Boyd, J.D., Bonds, A., Casagrande, V.A.: A comparison of koniocellular, magnocellular and parvocellular receptive field properties in the lateral geniculate nucleus of the owl monkey (aotus trivirgatus). The Journal of physiology **531**(1), 203–218 (2001) 3
44. Yang, C., Xu, Y., Shi, J., Dai, B., Zhou, B.: Temporal pyramid network for action recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 591–600 (2020) 4, 11, 12, 13
45. Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T.J., Lipson, H.: Understanding neural networks through deep visualization. ArXiv **abs/1506.06579** (2015) 7
46. Zamora Esquivel, J., Cruz Vargas, A., Lopez Meyer, P., Tickoo, O.: Adaptive convolutional kernels. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019) 4
47. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014) 7
48. Zhang, C., Gupta, A., Zisserman, A.: Temporal query networks for fine-grained video understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4486–4496 (2021) 2, 4, 11, 12, 13
49. Zhang, D.J., Li, K., Chen, Y., Wang, Y., Chandra, S., Qiao, Y., Liu, L., Shou, M.Z.: Morphmlp: A self-attention free, mlp-like backbone for image and video. arXiv preprint arXiv:2111.12527 (2021) 11
50. Zhang, J., Wang, Y., Zhou, Z., Luan, T., Wang, Z., Qiao, Y.: Learning dynamical human-joint affinity for 3d pose estimation in videos. IEEE Transactions on Image Processing **30**, 7914–7925 (2021) 4
51. Zhou, B., Andonian, A., Oliva, A., Torralba, A.: Temporal relational reasoning in videos. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 803–818 (2018) 4
52. Zhou, Y., Ni, B., Hong, R., Wang, M., Tian, Q.: Interaction part mining: A mid-level approach for fine-grained action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3323–3331 (2015) 2, 4