Google Helps YouTube: Learning Few-Shot Video Classification from Historic Tasks and Cross-Domain Sample Transfer

Xinzhe Zhou, Yadong Mu*
{zhouxinzhe1023,myd}@pku.edu.cn
Wangxuan Institute of Computer Technology, Peking University, China

ABSTRACT

The fact that video annotation is labor-intensive inspires recent research to endeavor on few-shot video classification. The core motivation of our work is to mitigate the supervision scarcity issue in this few-shot setting via cross-domain meta-learning. Particularly, we aim to harness large-scale richly-annotated image data (*i.e.*, source domain) for few-shot video classification (*i.e.*, target domain). The source data is heterogeneous (image v.s. video) and has noisy labels, not directly usable in the target domain. This work proposes *meta-learning input-transformer* (MLIT), a novel deep network that tames the noisy source data such that they are more amenable for being used in the target domain. It has two key traits. First, to bridge the data distribution gap between source / target domains, MLIT includes learnable neural layers to reweigh and transform the source data, effectively suppressing corrupted or noisy source data. Secondly, MLIT is designed to learn from historic video classification tasks in the target domain, which significantly elevates the accuracy of the unseen video category. Comprehensive empirical evaluations on two large-scale video datasets, ActivityNet and Kinetics-400, have strongly shown the superiority of our proposed method.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision tasks**; • **Information systems** → *Information systems applications.*

KEYWORDS

few-shot learning; meta-learning; video classification; cross-domain transfer

ACM Reference Format:
Xinzhe Zhou, Yadong Mu. 2020. Google Helps YouTube: Learning Few-Shot Video Classification from Historic Tasks and Cross-Domain Sample Transfer. In *2020 International Conference on Multimedia Retrieval (ICMR'20), June 8–11, 2020, Dublin, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3372278.3390687

*corresponding author.

1 INTRODUCTION

Deep learning has revolutionarily re-calibrated the state-of-the-art of many research tasks in the domains of computer vision, natural language processing and robotics etc. Despite its empirical success, some open problems remain unsolved. One of the most notorious ones is the data-hungry issue. Typically, gigantic meticulously-annotated data is required to ensure the learned model's generalization performance. Considering that annotating data can be intolerably labor-, money- and time-consuming in many tasks, learning from small samples (known as few-shot learning in the literature) becomes a technique that requires urgent exploration.

This work addresses few-shot video classification. Since video annotation demands at least a full-pass browsing of inspected videos, the scarcity of annotated data becomes particularly severe in video-oriented research. Although some large-scale video benchmarks have been recently established (such as Kinetics [11]), their tremendous annotating cost implies that this strategy cannot be trivially applied to other novel video categories. Our work is distinguished from exiting few-shot video classification methods by exploring two insights:

First, a majority of previous relevant efforts (such as compound memory network in [41]) only use very limited training samples and focus on maximally squeezing useful information from them. We here advocate a different approach of using cross-domain data. For example, web images can be semi-automatically annotated by their surrounding text. This sets up a massive albeit noisily-labeled data domain (hereafter referred to as the source domain). To overcome the domain discrepancy issue, we develop *meta-learning input-transformer* (MLIT), which aims to tame the source data such that they are more amenable for effectively augmenting the original few-shot data in the target domain. Specifically, MLIT jointly re-weighs each sample in the source / target domain based on its relative importance and mutual complementariness.

Secondly, some early work on few-shot learning [16] revealed the importance of taking advantage of previously-learned tasks when learning a new task. More recent progress elaborates on a novel learning-to-learn paradigm [4, 25, 38], dubbed as meta-learning. The learned model in meta-learning, called meta-learner, tunes part of its parameters concurrently by all historic relevant tasks and optimizes the rest specially for the current task. It is task-dependent to design the specific module whose parameters are used to "memorize" historic tasks. The module could be an optimizer [25], a scheme of parameter initialization [4], or a predictor for learner's weights [38]. In our proposed MLIT, a novel class-wise memory module and a transformer module are where previous video classification tasks pilot the re-weighing of all source / target data in a new task.

The proposed MLIT utilizes meta-learning for cross-domain data transfer. On the one hand, data from another source domain, transformed by MLIT, effectively mitigates the data scarcity issue in few-shot learning. On the other hand, the meta-learning framework enables learning from relevant tasks, which further elevates the performance. We conduct comprehensive experiments on two large-scale video classification datasets, ActivityNet [9] and Kinetics-400 [11]. All results strongly show that our model re-calibrates the state-of-the-art of few-shot video classification. We also provide various ablation study for algorithm analysis.

2 RELATED WORK

Few-shot Learning: The early development of few-shot learning can at least date back to [20]. Later, [14, 16] both devised generative models for one-shot learning. Very recently, the renaissance of deep neural networks [13] spurred various few-shot learning systems comprised of neural layers. For example, learning task-dependent metric from few samples exhibited strong performance when deep networks were adopted as the backbone of the model, including Siamese network [12], matching network [34], prototypical network [28], relation network [30] etc. Early thoughts about enhancing the very limited training data can be exemplified by hallucinating additional training examples for data-starved classes [6], or harvesting auxiliary machine-labeled web images for cross-domain object recognition [39].

Meta-learning: The concept of meta-learning or learning-to-learn was previously developed in [22, 26, 31]. Unlike conventional supervised learning, meta-learning treats a learning task (with its own training and testing data) as an individual “sample". It aims to learn a meta-learner on a group of tasks, such that the learned meta-leaner can adapt to a new task with least effort and be more overfitting-resistant. To this end, part of its parameters are designed to “memorize" historic tasks, and the rest are specially optimized for a new task. The module used to “memorize" historic tasks varies across different tasks. For example, the module could be an optimizer [1, 17, 25], a scheme of parameter initialization [4], a predictor for learner’s weights [38], a collection of optimization settings (e.g., initial parameters, update direction and learning rate) [18]. In [21], authors designed this module as a learnable black-box function instead of a hand-crafted one.

Video Classification: It has been a long-standing research task and recently revolutionized by deep networks. Early works used hand-designed features like STIP [15] and improved dense trajectories (iDT) [35]. Recent research has emphasized end-to-end deep networks, particularly 3-D convolutional CNN [2, 24, 32], recurrent networks [23, 40], and multi-stream fusion [27, 36]. All aforementioned methods suffer from the intrinsic data-hungry issue of deep models. Few-shot video classification, which is the main scope of this work, aims to relieve the data requirement and obtain good model performance with few training videos.

3 APPROACH

3.1 Task Settings

Our formulation for few-shot video classification elegantly unifies both cross-domain data transfer and learning from historic

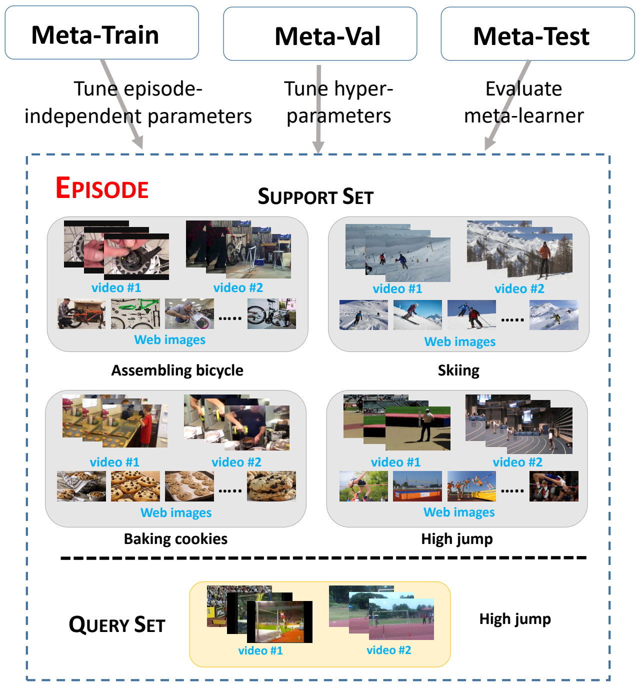

Figure 1: Illustration of an episode of 4-way 2-shot meta-learning.

tasks under meta-learning framework. Formally, we have three meta sets called meta-training, meta-validation and meta-testing set respectively. The role of meta-sets is essentially akin to the train / validation / test sets in standard supervised learning, but the “samples" in meta-sets are few-shot classification tasks rather than individual videos. Training a meta-learner is accomplished by performing a series of n-way k-shot *episodes* (analogous to an optimization iteration in standard supervised learning). To start a new episode, the algorithm samples n categories from one meta-set, with k videos selected for each of them, forming a *support set* of $n \times k$ samples. Likewise we construct another *query set* for evaluating models adapted on the support set. Figure 1 depicts a typical 4-way 2-shot episode in few-shot video classification.

A meta-learner is expected to fully memorize category-agnostic knowledge from episodes sampled from meta-training, and generalize well to episodes in meta-testing with novel categories. To this end, the optimization of a meta-learner is performed as follows: we first optimize the loss on the support set to refine all episode-dependent parameters. Then the loss on the query set is calculated, but the corresponding gradient is only used to update all episode-independent parameters (which encode common knowledge for all categories). When learning never-seen categories from meta-testing, the episode-independent parameters are frozen, and the rest are optimized via the support set. Performance on the query set is recorded for each episode, and averaged accuracy on all episodes in meta-testing are taken as the generalization performance of the meta-learner. The whole process is described in Algorithm 1.

Our model exploits other cross-domain data sources that are imperfect but plentiful. Particularly, we treat each video category

Algorithm 1 Cross-Domain Meta-Learning for Few-Shot Video Classification

```
1: Input: data sets including meta-train $\mathcal{D}_{meta\_train}$, meta-val $\mathcal{D}_{meta\_val}$ and meta-test $\mathcal{D}_{meta\_test}$, all of which are augmented by webly-crawled images;
2: Parameters: FC layers in feature extractor; LSTM in MLIT; hyper-parameters (visual feature dimension, memory size and $T$ in MLIT etc.);
3: Output: Feature extractor; LSTM; down-stream classifier;
   Phase of Meta-Training
4: for each episode do
5:     Sample $n$ video classes from $\mathcal{D}_{meta\_train}$ each with $k$ examples in support and some others as query, creating a $n$-way $k$-shot episode;
6:     (On the support set) Run feature extractor; Insert feature vectors into the memory module of MLIT;
7:     (On the support set) Run transformer module and obtain $w$ for each frame and web image; Calculate class prototypes by weighted summation of support features;
8:     (On the query set) Extract video feature; Calculate video classification loss and perform gradient descent to update model parameters;
9: end for
   Phase of Meta-Validation
10: for each episode do
11:     Sample from $\mathcal{D}_{meta\_val}$ and create a $n$-way $k$-shot episode;
12:     Adapt task-dependent parameters on support set; Calculate and record video classification loss on query set;
13: end for
14: Select optimal hyper-parameters;
   Phase of Meta-Testing
15: for each episode do
16:     Sample from $\mathcal{D}_{meta\_test}$ and create a $n$-way $k$-shot episode;
17:     Adapt task-dependent parameters on support set; Calculate and record video classification loss on query set;
18: end for
19: Calculate average accuracy;
```

name as keywords (with proper processing, such as converting "passing American football (not in game)" to "passing American football -game") and retrieve top-ranked images from Google's image search engine. Note that webly-crawled images are only required in the support set since classifying videos in the query set does not rely on external information, as shown in Figure 1.

The proposed model is comprised of three components: feature extractor, *meta-learning input-transformer* (MLIT) and down-stream classifier. Following a common practice in few-shot learning, the classifier we adopt is a prototypical one, namely weighted average of all training samples (including web images and video frames) from specific video category. A novel video will be classified to the closest prototype's class. To update model parameters, classification loss needs to be calculated on the query set, which roots the gradient to be back-propagated. Suppose video $\mathbf{x}$ belongs to class c among all n video classes in an episode. Inspired by [41], we use the loss function below:

$$\mathcal{L}(\mathbf{x}, c) = \max(\alpha - s_c(\mathbf{x}) + \sup_{k \in \{1 \dots n\} \backslash c} s_k(\mathbf{x}), 0), \tag{1}$$

where $s_c(\mathbf{x})$ is the classifier confidence of class c for video $\mathbf{x}$ and α is a hyper-parameter that defines the level of margin between positive / negative classes.

3.2 Feature Extractor

Importantly, the image / video domains are intrinsically heterogeneous. To fairly compare with competitors (such as CMN [41]), we simply follow the common practice that ignores the video temporal dynamics and treat each video as a loose set of frames. All video frames and webly-crawled images are fed into a feature extractor.Video frames and images first go through some pre-trained deep model (we adopt ResNet50 [7, 8]). The feature map of the penultimate layer is extracted and flattened into a 2,048-dimensional vector. A learnable fully-connected (FC) layer is followed to tune the general features to be more task-specific.

3.3 Meta-Learning Input-Transformer (MLIT)

The core of our model, which we call as *meta-learning input-transformer* (MLIT), is a new kind of meta-learner characterized with cross-domain data transfer and memory-based meta learning. It is comprised of a *memory module* and a *transformer module*, as illustrated in Figure 2.

3.3.1 Class-wise Memory Module. Memory module [5, 10, 33, 37] is one of the research front in few-shot learning. It resembles a physical storage to store and fetch data. Modern representative memory modules include end-to-end memory network [29] and key-value memory network [19] etc. Previous works often use one single memory where stored data from different classes are unsorted. In order to model the class-aware context more conveniently, we devise a dexterous memory module which stores different-class data at isolated parts. Its architecture is found in Figure 2. For statement clarity, let us denote the whole memory as Mem and sub-part for class c as Mem_c. $c = 1 \dots n$ for n-way episode.

In all meta-learning phases, we first extract features of video frames in the support set, and store each of them in corresponding memory part Mem_c based on its class c (assume frames inherit the video label). Web images are excluded from updating Mem considering their inherent noises, thus unsuitable to serve as bases of attention model as explained in Equations (3)(5). Given limited slots in each Mem_c, we pre-select a random sub-set frames from the query set in order to fit the capacity of all Mem_c.

3.3.2 Attentional Transformer Module. The transformer module in Figure 2 is based on LSTM and class-wise attention model. Its main function is to re-weigh cross-domain data, such that most informative data will be assigned higher weight in calculating class prototype. The controller logic of transformer module is defined by

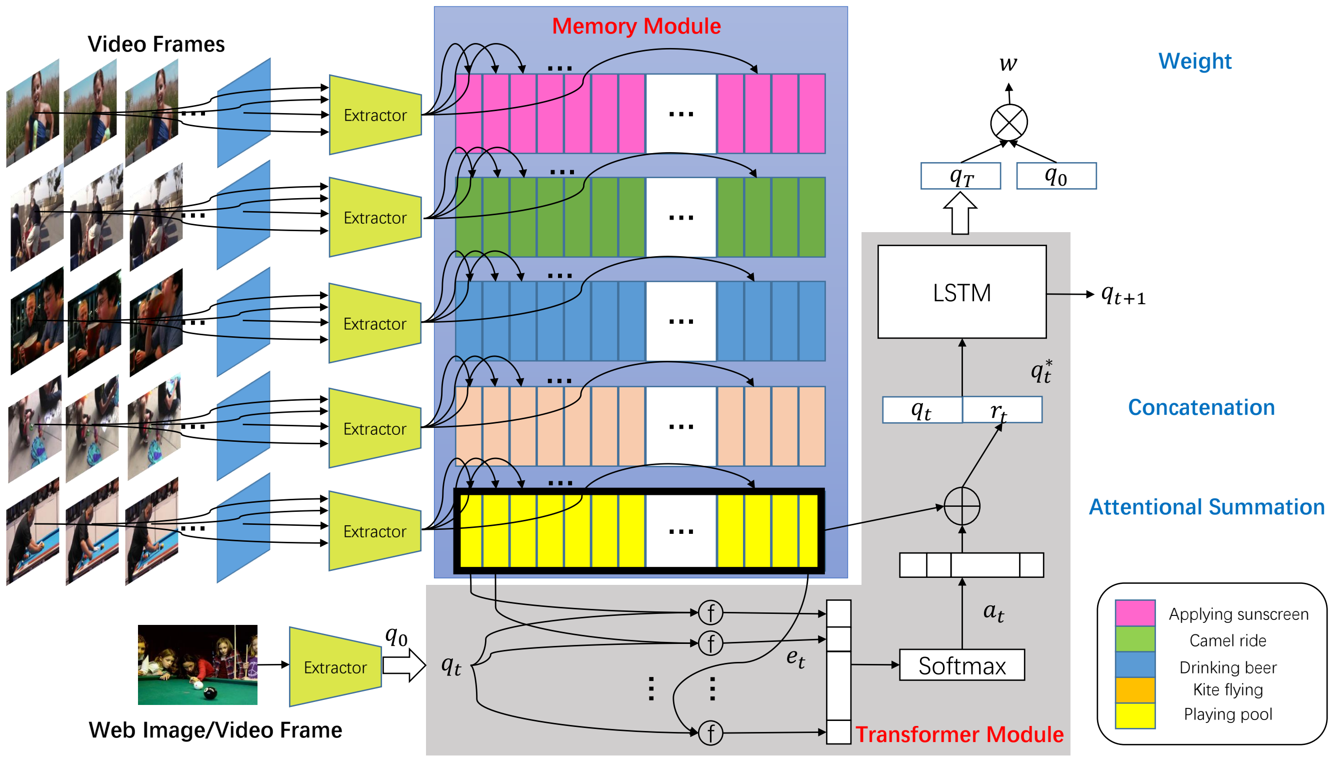

Figure 2: Architecture of meta-learning input-transformer (MLIT). Better viewing in color mode.

the formulas below:

$$(LSTM):q_t = LSTM(q_{t-1}^*), \tag{2}$$

$$(Attention):e_{i,t} = \psi(m_i)^\top \psi(q_t), \tag{3}$$

$$a_{i,t} = \frac{\exp(e_{i,t})}{\sum_j \exp(e_{j,t})}, \tag{4}$$

$$(Attentional\ summation):r_t = \sum_i a_{i,t} m_i, \tag{5}$$

$$(Concatenation):q_t^* = [q_t\ r_t], \tag{6}$$

$$(Weight):w = q_0^\top q_T, \tag{7}$$

where $\psi(\cdot)$ denotes some learnable vector-to-vector mapping function.

For an arbitrary input instance (could be either a video frame or web image), it first goes through the feature extractor as described in Section 3.2. Denote the feature vector as $q_0 \in \mathbb{R}^d$. The LSTM unit in Figure 2 is initialized by simple all-zero cell state, and gets recurrently updated according to Eqn. (2)-(6). This generates a sequence of hidden states $q_1, \ldots, q_T$, where T is a hyper-parameter to be tuned. Let c be the video class that this instance is known to be from or hypothesized. Critically, only slots in Mem_c are involved in the computations. Denote the vector fetched from the i-th slot of Mem_c as $m_i \in \mathbb{R}^d$. The computation at the recurrent iteration t starts from comparing q_t with each m_i, according to Eqns. (3)(4). The resultant $e_{i,t}$ are further normalized to obtain the *attention score* $a_{i,t}$, which are then used to attentionally sum the memory vectors to get an aggregated vector r_t. Intuitively, r_t encodes all relevant information from the memory module to currently-inspected instance. As seen in Eqn. (6), we finally concatenate r_t and q_t itself for feeding the LSTM unit, generating a new hidden state (also the input to the next recurrence) q_{t+1}.

Above recurrences do the job of progressively removing irrelevant information from the input instance. This module is richly parameterized. It is thus capable of learning an optimal module such that the final hidden state q_T after T steps preserves most of class-specific information. Intuitively, large gap between q_0 and q_T implies a high level of noise in the input. We multiply the two as Eqn. (7) to obtain a scalar w that plays a key role in computing down-stream classifiers.

3.4 Down-Stream Classifier

After all frames / web images in the support set are weighted according to Eqn. (7), we obtain a full collection of weights $w_{c,j}$ (j indexes the frame and web image) for each class c. A contextual weight can be further computed via $w_{c,j} \leftarrow \exp(w_{c,j})/\sum_j \exp(w_{c,j})$. A class-specific prototype p_c ($c = 1 \ldots n$ for all n video categories) is simply computed via weighted average, namely

$$p_c = normalize\left(\sum_j w_{c,j} \cdot q_0(j)\right), \tag{8}$$

where $q_0(j)$ denotes extracted feature of video j and *normalize*() ensures all prototypical vectors have unit norm.

When conducting classification on a querying video, we first extract all of its frame features, and then average and normalize all the features vectors to obtain a *video mean vector*. Denote it by $\bar{q}$. The label of this video is regarded to be the specific class that admits a maximal correlation, namely $\hat{c} = \arg\max_c \ \bar{q}^\top p_c$ where where $\hat{c}$ is the predicted video label.

4 EXPERIMENT

4.1 Data Description and Preparation

We experimented with two large-scale video benchmarks, whose information is given below:

ActivityNet (version 1.3) [9]: contains totally 19,994 videos from 200 rich semantic classes (such as "paining", "cutting the grass" etc.). We abandon the original testing set since the ground-truth labels are kept by data organizers. For the rest data, we randomly split them into 128, 24 and 48 non-overlapping classes as meta-train/val/test respectively. For each video category, the top-400 images crawled from Google image search engine are used as cross-domain data.

Kinetics-400 [11]: consists of 306,245 video clips from 400 categories. Videos have relatively shorter durations. Following [41], we randomly select 100 classes out of 400, and split them into 64, 12 and 24 non-overlapping classes for constructing meta-sets. We ensure all chosen classes have at least 150 videos. The first top-400 images crawled from Google image search engine are kept, same as above.

4.2 Competing Methods

Since we adopt episodic training and evaluation, it makes more fair comparisons between ours and meta-learning based methods. We thus primarily include recent meta-learning methods, as well as some traditional classifiers. The input for all baselines are pre-extracted features of web images and video frames (ResNet50 features in our experiments). All baselines are implemented in PyTorch, either by original authors or us.

Nearest Neighbour (NN): We implemented two versions of prototypical NN as follows: NN_1: for each class in the support set, it first averages all frame features of one video to obtain video-level representation, and then all video-level features are further averaged into a video prototype. All web images are also averaged into an image prototype. The video / image prototypes are eventually averaged and normalized to the prototype of current class. NN_2: almost the same to NN_1, with slight difference that the class prototypes are obtained by indiscriminately averaging all video frames / web images.

Linear Classifier (LC): is parameterized as $g(x) = Wx + b$. For each episode, we randomly initialize parameters W and b, then train the model using the support set for many iterations. After convergence, its performance is evaluated on query data. Two versions of LC are implemented: LC_{mix}: is trained using a hybrid of video frame and web image features. LC_{sep}: is pre-trained on web image features and then fine-tuned on video frame features.

Support Vector Machine (SVM) [3]: We include both LinearSVM and RBFSVM (using a standard RBF kernel).

Prototypical Network (ProtoNet) [28]: performs few-shot classification via learning prototypical representation for each class. We instantiate its embedding function with a 2048×2048 FC layer. Euclidean distance is used when comparing a query with prototypes.

Compound Memory Network (CMN) [41]: We implement CMN as the original paper described. Since CMN is designed to deal with videos (*i.e.*, frame sets), we enable it to tackle web images as follows: for every class, randomly sample K web images to form a pseudo-video's frame set. Repeat the sampling M times to produce M pseudo-videos. We empirically set $K = 100$ and $M = 5$ in the experiments.

4.3 Evaluation

We evaluate the model under 5-way 1-shot (*i.e.*, each episode contains 5 active classes, each of which has exactly one video example as in either the support or query set respectively) and 5-way 5-shot settings on both ActivityNet and Kinetics-400. The performance is reported in terms of average precision from multiple episodes, as shown in Table 1. Some observations can be drawn as below:

1. MLIT is not the top performer under the settings that no web data is involved. Without web images, because most video frames are semantically relevant, the weights learned by MLIT are nearly uniform. For example, for 5-way 1-shot meta-testing on ActivityNet, the averaged min / max weights over many episodes are 0.0200 and 0.0202 respectively, implying a near-zero weight variance.In such scenarios, our proposed method degrades to feature adaptation + nearest neighbor classification. The reported accuracies are thus reasonably in-between NN and other more sophisticated methods (such as CMN, which is specially designed for video temporal structure).

2. When supplemented with extra web images, MLIT consistently shows its superiority by out-performing all baselines, except for the 5-shot setting on Kinetics-400. We attribute the non-trivial performance promotion to web image re-weighing conducted at the transformer module of MLIT. In contrast, most of other competing methods (except for CMN) indiscriminately use noise web images. We can also find indirect cues of MLIT's effectiveness by investigating the min / max sample weights averaged over multiple meta-testing episodes. For example, for 5-way 1-shot learning on ActivityNet, the averaged min / max weights for web images are 0.0001 and 0.0339 respectively, implying some web images receive higher importance after passing the transformer module of MLIT and other noisy ones are suppressed. Visualization of cross-domain sample weights are deferred to Section 4.5.

3. For 5-way 5-shot with web images on Kinetics-400, MLIT still beats all other meta-learning based methods by large margins. However, the traditional method RBFSVM tops all the others. In fact, a close investigation reveals that the frame count of a single video in Kinetics-400 is significantly larger than that in ActivityNet (roughly 300 v.s. 100 frames, here we conducted temporal subsampling for ActivityNet beforehand), making the situation similar to large-shot learning. Consequently, non-meta-learning methods take the advantage through adequate supervision.

Table 1: Average accuracy of 5-way 1-shot and 5-way 5-shot on the meta-testing set of ActivityNet (in top table) and Kinetics-400 (in bottom table). The accuracies are reported in the range of [0,100].

Model	1-shot w/o web	1-shot w/ web	5-shot w/o web	5-shot w/ web
NN_1	64.84	81.14	86.26	88.28
NN_2	64.84	83.16	84.36	87.84
LC_{mix}	61.4	86.48	83.7	90.2
LC_{sep}	-	83.3	-	88.72
LinearSVM	60.0	86.38	82.0	89.52
RBFSVM	54.52	85.06	81.9	89.36
ProtoNet	71.52	86.0	85.10	87.94
CMN	**78.64**	88.04	**88.58**	90.46
MLIT(ours)	68.14	**88.96**	86.8	**90.76**

Model	1-shot w/o web	1-shot w/ web	5-shot w/o web	5-shot w/ web
NN_1	59.26	75.04	77.66	81.32
NN_2	59.26	77.44	77.0	80.12
LC_{mix}	55.14	81.06	77.16	82.9
LC_{sep}	-	78.84	-	81.88
LinearSVM	56.76	78.72	73.52	81.68
RBFSVM	48.96	80.48	74.94	**83.32**
ProtoNet	63.38	76.96	76.14	78.78
CMN	**66.58**	80.5	77.7	80.94
MLIT(ours)	63.74	**81.3**	**79.44**	81.84

Table 2: Noise-resistant experimental results on ActivityNet (in top table) and Kinetics-400 (in bottom table) under 5-way 1-shot settings. The reported are averaged precision in the testing episodes. The values in the parentheses are the relative accuracy promotion with respect to the baseline whose performance is closest to MLIT's.

Model	0	0.1	0.3	0.5	1.0
NN_1	81.14	80.86	79.98	78.5	69.54
NN_2	83.16	83.72	84.76	84.6	61.1
LC_{mix}	86.48	86.92	84.62	82.02	42.3
LC_{sep}	83.3	82.04	78.64	76.2	46.88
LinearSVM	86.38	84.48	80.12	75.48	27.98
RBFSVM	85.06	85.08	84.78	83.58	37.6
ProtoNet	86.0	85.40	85.32	84.34	60.52
CMN	88.04	87.98	87.22	87.28	73.02
MLIT(ours)	**88.96**(+0.92)	**88.36**(+0.38)	**88.2**(+0.98)	**87.52**(+0.24)	**76.86**(+3.84)

Model	0	0.1	0.3	0.5	1.0
NN_1	75.04	74.56	72.8	70.24	61.5
NN_2	77.44	77.38	76.22	74.3	60.4
LC_{mix}	81.06	**81.18**	78.66	74.58	35.78
LC_{sep}	78.84	78.46	76.4	73.86	51.24
LinearSVM	78.72	76.18	68.42	61.32	20.32
RBFSVM	80.48	80.54	79.48	79.36	21.28
ProtoNet	76.96	76.72	75.64	73.66	60.92
CMN	80.5	78.94	78.14	77.44	62.46
MLIT(ours)	**81.3**(+0.24)	80.7(-0.48)	**80.58**(+1.1)	**79.46**(+0.1)	**66.16**(+3.7)

4.4 Ablative Study

The major advantage of MLIT is its noise-resist property to cross-domain data noise. We there conduct two ablative experiments regarding the effect of denoising.

Effect of Noise Level: To compare the robustness under varying noise levels, we manually control the noise level in web images as follows: Given a target noise level $p \in [0, 1]$ and assume there are totally N_{img}^c web images in class c, we randomly choose $p \times N_{img}^c$ images from class c, and replace them with some random images from other classes. Obviously, $p = 0$ boils down to the original setting and $p = 1.0$ implies fully noisy cross-domain data. With above-mentioned synthetic noisy data, we conduct comparative studies under the 1-shot setting.

Table 3: Average accuracy for MLIT with (marked by ✓) and without (marked by ✗) transformer module on ActivityNet (in top table) and Kinetics-400 (in bottom table). Values in the parentheses are the relative promotion.

w/ transformer	1-shot w/o web	1-shot w/ web	5-shot w/o web	5-shot w/ web
✗	68.02	87.94	86.74	90.72
✓	68.14(+0.12)	88.96(+1.02)	86.8(+0.06)	90.76(+0.04)

w/ transformer	1-shot w/o web	1-shot w/ web	5-shot w/o web	5-shot w/ web
✗	63.5	79.9	79.08	80.76
✓	63.74(+0.24)	81.3(+1.4)	79.44(+0.36)	81.84(+1.08)

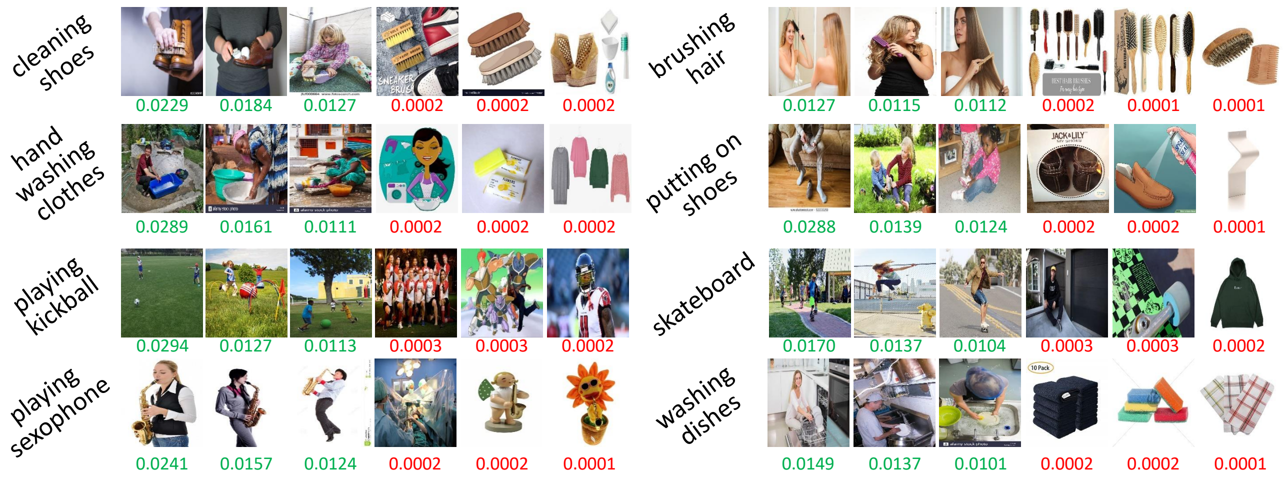

Figure 3: Some typical image-weight pairs generated by MLIT. The left four classes are drawn from the meta-train set, and the right four are from the meta-test sets. Sample weights are plotted under corresponding video frames or web images. Specifically, high-weight / low-weight images are shown in green / red color respectively.

Table 2 summarizes the experimental results on ActivityNet and Kinetics-400 respectively. As seen, on ActivityNet our proposed MLIT dominates at all noise levels and performs best at a majority of noisy levels on Kinetics-400. Remarkably, the performance gap between MLIT and the second best baseline tends to become larger as the noise levels grow. Under the extreme condition of $p = 1.0$, MLIT surpasses the second best model CMN with large gaps of 3.84% and 3.7% for the two datasets, respectively. This faithfully demonstrates MLIT's ability of suppressing cross-domain noises and distilling useful information.

Effect of Transformer Module: As the core of our model, the transformer module learns to denoise cross-domain data. To quantitatively illustrate the benefit of including the transformer module in the pipeline, we also experiment with or without (namely only feature adaptation is kept) this module under various settings. Table 3 summarizes the comparisons on ActivityNet and Kinetics-400 respectively. Importantly, the performance gains are notably larger when web data is included. It strongly explains the effectiveness of our proposed transformer module.

4.5 Visualization of Sample Weighting

As stated before, video frames and web images are mixed and fed into the transformer module. A natural way for investigating the data denoising effect is via checking real images and the associated weights assigned by the transformer. To this end, we select some typical image-weight pairs from both meta-train and meta-test sets and make visualization in Figure 3. It can be intuitively observed that semantically relevant / irrelevant images are discriminatively weighed, which further corroborates the effectiveness of MLIT.

5 CONCLUDING REMARKS

This work tackles the data scarcity issue inherent in few-shot video classification by harnessing large-scale richly-annotated image data from the Web. The proposed model, *meta-learning input-transformer* (MLIT), follows the idea of meta-learning, while utilizes a novel input transformer module to suppress corrupted or noisy data, and extract useful information to supplement the supervision. Experiments on two large-scale datasets, ActivityNet and Kinetics-400, clearly shows the superiority of our proposed method. More analysis and interference experiments further prove the effectiveness and robustness of MLIT.

ACKNOWLEDGMENTS

This work is supported by National Key R&D Program of China (2018AAA0100702), Beijing Natural Science Foundation (Z190001) and National Natural Science Foundation of China (61772037).

REFERENCES

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. 2016. Learning

to learn by gradient descent by gradient descent. In *NIPS*.

[2] João Carreira and Andrew Zisserman. 2017. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *CoRR* abs/1705.07750 (2017).

[3] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Mach. Learn.* 20, 3 (1995), 273–297.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

[5] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *CoRR* abs/1410.5401 (2014).

[6] Bharath Hariharan and Ross B. Girshick. 2017. Low-Shot Visual Recognition by Shrinking and Hallucinating Features. In *ICCV*.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *ECCV*.

[9] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. ActivityNet: A large-scale video benchmark for human activity understanding. In *CVPR*.

[10] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to Remember Rare Events. In *ICLR*.

[11] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. 2017. The Kinetics Human Action Video Dataset. *CoRR* abs/1705.06950 (2017).

[12] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICMLW*.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.

[14] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. One shot learning of simple visual concepts. In *CogSci*.

[15] Ivan Laptev. 2005. On Space-Time Interest Points. *International Journal of Computer Vision* 64, 2-3 (2005), 107–123.

[16] Fei-Fei Li, Robert Fergus, and Pietro Perona. 2006. One-Shot Learning of Object Categories. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (2006), 594–611.

[17] Ke Li and Jitendra Malik. 2017. Learning to optimize. (2017).

[18] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-SGD: Learning to Learn Quickly for Few Shot Learning. *CoRR* abs/1707.09835 (2017).

[19] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

[20] Erik G. Miller, Nicholas E. Matsakis, and Paul A. Viola. 2000. Learning from One Example through Shared Densities on Transforms. In *CVPR*.

[21] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A Simple Neural Attentive Meta-Learner. In *ICLR*.

[22] Devang K Naik and RJ Mammone. 1992. Meta-neural networks that learn by learning. In *IJCNN*.

[23] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *CVPR*.

[24] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning Spatio-Temporal Representation with Pseudo-3D Residual Networks. In *ICCV*.

[25] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

[26] Jürgen Schmidhuber. 1987. *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook.*

[27] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *NIPS*.

[28] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *NIPS*.

[29] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *NIPS*.

[30] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to Compare: Relation Network for Few-Shot Learning. In *CVPR*.

[31] Sebastian Thrun and Lorien Pratt. 1998. *Learning to learn.* Springer.

[32] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *ICCV*.

[33] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order Matters: Sequence to sequence for sets. In *ICLR*.

[34] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *NIPS*.

[35] Heng Wang and Cordelia Schmid. 2013. Action Recognition with Improved Trajectories. In *ICCV*.

[36] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. *CoRR* abs/1608.00859 (2016).

[37] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *ICLR*.

[38] Tailin Wu, John Peurifoy, Isaac L. Chuang, and Max Tegmark. 2018. Meta-learning autoencoders for few-shot prediction. *CoRR* abs/1807.09912 (2018).

[39] Zhongwen Xu, Linchao Zhu, and Yi Yang. 2017. Few-Shot Object Recognition from Machine-Labeled Web Images. In *CVPR*.

[40] Linchao Zhu, Zhongwen Xu, and Yi Yang. 2017. Bidirectional Multirate Reconstruction for Temporal Modeling in Videos. In *CVPR*.

[41] Linchao Zhu and Yi Yang. 2018. Compound Memory Networks for Few-Shot Video Classification. In *ECCV*.