Label Independent Memory for Semi-Supervised Few-shot Video Classification

Linchao Zhu, Yi Yang

Abstract—In this paper, we propose to leverage freely available unlabeled video data to facilitate few-shot video classification. In this semi-supervised few-shot video classification task, millions of unlabeled data are available for each episode during training. These videos can be extremely imbalanced, while they have profound visual and motion dynamics. To tackle the semi-supervised few-shot video classification problem, we make the following contributions. First, we propose a label independent memory (LIM) to cache label related features, which enables a similarity search over a large set of videos. LIM produces a class prototype for few-shot training. This prototype is an aggregated embedding for each class, which is more robust to noisy video features. Second, we integrate a multi-modality compound memory network to capture both RGB and flow information. We propose to store the RGB and flow representation in two separate memory networks, but they are jointly optimized via a unified loss. In this way, mutual communications between the two modalities are leveraged to achieve better classification performance. Third, we conduct extensive experiments on the few-shot Kinetics-100, Something-Something-100 datasets, which validates the effectiveness of leveraging the accessible unlabeled data for few-shot classification.

Index Terms—Few-shot Video Classification, Semi-supervised Learning, Memory-augmented Neural Networks, Compound Memory Networks

1 Introduction

To successfully train a deep network, millions of labeled video examples are required. However, in some real-world scenarios, it is unrealistic to manually collect large datasets for a new task. Thus, it gives rise to few-shot classification, where the goal is to quickly generalize to a novel task from only a few training examples. Recently, few-shot classification has attracted considerable research interests [1], [2], [3], [4], [5], focusing on image classification using metric-learning [1], [2] and gradient-based optimization [4]. In many few-shot problems, few-shot video classification [6] is another important task, aiming to enable the agent to quickly learn and understand surroundings from a few sequencial observations. Understanding the video content from a few examples is a more challenging task, however, less attention has been paid to it. Videos have more complex structures than images, involving temporal information and more noise, such as camera motion, object scale and viewpoint variances. Many videos contain hundreds of frames with complex scene dynamics. With this complexity, it may be difficult to understand the concept in a video when only a few examples are provided.

Few training examples hinders the model when learning a discriminative video representation. One of the possible solutions for few-shot video classification is to leverage a large amount of freely accessible unlabeled data. When large amount of video data are introduced for few-shot training, it is essential to model the unlabeled video distributions and extract appropriate video representations. This process, together with the training loss on the few labeled video data sets, can be difficult to optimize. In [7], Hsu et al. proposed to train a general embedding function from the unlabeled data first, to be used as initialization for few-shot training. The unsupervised training process is not conditional on any target data, which limits the generalization. It is also difficult to learn a universal feature extractor from unlabeled data.

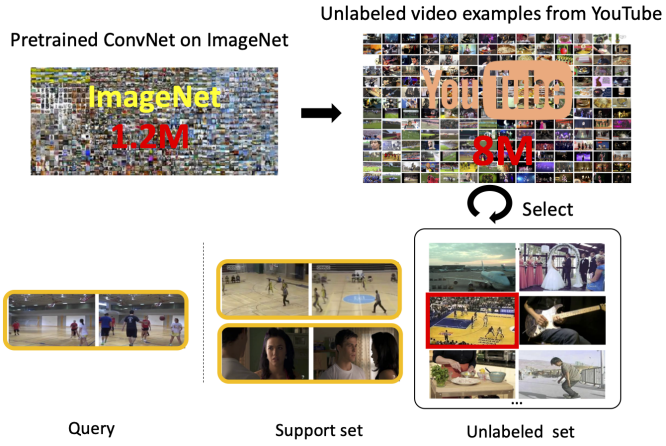

Fig. 1: We leverage off-the-shelf feature extractors trained on ImageNet to extract frame-level embeddings for unlabeled videos. The related videos will be selected from a large video dataset to form an unlabeled set. This unlabeled set will then be utilized for few-shot training.

As an alternative approach, we leverage an existing off-the-shelf feature extractor for the unlabeled video data, to ease the learning difficulties (Fig. 1). Fortunately, the modern convolutional networks trained on ImageNet show attractive transferability properties. Instead of learning representation from the unlabeled data from scratch, the existing convolutional models can serve as universal feature extractors for video frames on novel categories. YouTube-8M [8] contains millions of videos from various of domains.

• *Linchao Zhu and Yi Yang are with University of Technology Sydney (email: linchao.zhu@uts.edu.au and yi.yang@uts.edu.au). Yi Yang is the corresponding author.*

To generate the frame-level representations, the authors simply extract the Inception-V3 [9] features, where the model is pre-trained on ImageNet. These features are great sources of profound visual and motion dynamics. In this paper, we propose to leverage the massive off-the-shelf video features for few-shot video classification. However, we do not use any labeled information to setup more realistic semi-supervised few-shot settings. In addition to leveraging unlabeled data for few-shot video classification, we make the following contributions.

First, a label independent memory (LIM) bank is proposed to cache class specific knowledge. To leverage the unlabeled data for few-shot video classification, we propose to assign labels to the unlabeled data. However, instead of assigning the class label to each individual data point, we introduce a label independent memory bank to store the relatedness between the unlabeled data and the target examples. The label will be assigned to the class prototype, which is a weighted average of the features in each LIM bank. In addition, we introduce the read and the write operations upon LIM. The class prototype can relieve the noise caused by inaccuracies when selecting of unlabeled data.

Second, we introduce two separate episodic memory banks to individually store the RGB information and the motion information. Instead of storing the RGB and optical flow representation in the same memory, we introduce two structures for storing different cues. Based on the proposed Compound Memory Network (CMN) [6], we introduce a unified loss to train the embedding function for the target datasets. Though the two-stream features are stored independently, the unified loss will build connections between the two CMNs, enabling their mutual communication. The original CMN structure is designed based on the key-value memory networks [10]. During training, information in each training episode is gradually accumulated into CMN.

Third, equipped with the above components, we achieve state-of-the-art performance on the few-shot Kinetics-100 dataset [6]. To avoid overfitting on this dataset, we additionally collect few-shot Something-Something-100 dataset. Extensive experiments validates the effectiveness of each component. With multi-modality CMN, we observe a significant performance gain on both datasets. Our work also allows for future research on the semi-supervised few-shot video classification problem.

2 Related Work

2.1 Video classification

The success of CNNs in image understanding [11], [12], has been useful for various video understanding tasks, including action recognition [13], action detection [14], video captioning [15]. Video classification methods have evolved from using hand-crafted features, e.g., improved dense trajectories [16], to deep models, e.g., two-stream Convolutional Neural Networks (ConvNets) [13], [17], 3D ConvNets [18], two-stream 3D ConvNets [19]. Some research has been conducted on encoding deep features to a global representation. For instance, Arandjelović *et al.* [20] proposed a NetVLAD layer for image retrieval and achieved improvements over unsupervised VLAD. It is later been used in [21] and [22] for video data modeling. These VLAD-based methods focus on video-level feature encoding via adaptive center assignment. Our multi-saliency embedding function leverages hidden descriptors and the assignment weights are learned with attention mechanism. Unsupervised video pre-training and multi-modal video text pretraining have been studied in [23], [24], and we mainly focus on semi-supervised video classification. After the releasing of the Kinetics dataset [19], more 3D convolutional networks have been proposed [25], [26]. These efforts have been made to train a video classification model using large amounts of video data, however, it would be expensive to collect large datasets and retrain the network for all novel tasks. The few-shot video classification task is more realistic in a real-world scenario, where the model will encounter novel categories that do not appear in the training process. The networks should learn to adapt to new tasks.

2.2 Few-shot Representation Learning

Early works from Miller et al. [27], Fei-Fei et al. [28] and Lake et al. [29] utilized generative models with one-shot learning. Santoro et al. [30] was the one of the first works to successfully integrate memory networks to few-shot learning. Vinyals et al. [1] used metric learning for few-shot recognition. The network is trained to find the nearest instance in the support set, then the corresponding label is retrieved. Snell et al. [2] utilized a prototype representation to stabilize the matching performance, and they used Euclidean distance with their embedding function. Finn et al. [4] used gradient-based update (SGD) as a meta-learner to optimize the learner's parameters. These few-shot learning methods target at image classification, where the studies on few-shot video classification are largely neglected. The existing image components cannot be used directly to model temporal dynamics in videos. Recently, few-shot video classification [6], [31] has been proposed to address a more difficult problem, which requires to capture motion dynamics from a few video clips. In this paper, we further introduce a promising direction to improve few-shot video classification accuracy by leveraging freely available unlabeled data. We dynamically select relevant examples from unlabeled data. The relevant examples are stored to a label independent memory for better few-shot video classification.

2.3 Semi-supervised Few-shot Learning

Semi-supervised learning has achieved great progress in recent years [32], [33], [34]. These methods usually leverage unlabeled data to alleviate the need of collecting sufficient labeled data. Laine et al. [32] proposed a temporal ensembling method to maintain an exponential moving average of label predictions on each training sample, where the predictions that are inconsistent with the target will be penalized. This consistency regularization helps to learn a better predictor for the unknown labels. Mean Teacher [33] improves temporal emsembling by leveraging exponential moving average on convolutional weights instead of label predictions. Berthelot et al. [34] introduced an effective MixMatch algorithm that predicts labels for the unlabeled examples. Later, MixUp [35] is used to mix labeled and

unlabeled data.

In semi-supervised few-shot image classification, [36] used well-constructed unlabeled data to update the original prototype, and utilized a soft k-means for cluster center tuning on the unlabeled data. However, they leveraged unlabeled data that are specifically given at each episode, and the unlabeled data usually consist of small number of examples. Liu et al. [37] tackled few-shot learning in the transductive setting, which learns to model the distributions of the testing data and propagates labels from labeled data to unlabeled data. Li et al. [38] proposed to initialize a self-training model for cherry-picking examples from noisy labels. In this paper, we leverage the unlabeled data in the feature space and focus on selecting related videos for target training. We exploit a nearest neighbour method to select examples from unlabeled data. The similarity scores are stored in a label independent memory to enhance the robustness of label prediction.

3 Semi-supervised Few-shot Video Classification

In this section, we show that CMN can be readily applied to semi-supervised few-shot video classification. Few-shot video classification is still a relatively new task. Unlike few-shot image classification, one of the challenges is for the model to learn the video dynamics from a few examples. Given that the model is provided with a few examples, it becomes more difficult to recognize objects, as well as motion changes in high dimensional data, although it is promising to incorporate large volumes of unlabeled data for few-shot video classification. In image classification, semi-supervised few-shot classification has been proposed, where unlabeled examples are included within each episode [36]. However, in real-world scenarios, structured unlabeled data are often difficult to collect. A more common method would be to enable the model to access all the unlabeled data during the training process. More recently, Hsu et al. [7] proposed to learn image embeddings from unlabeled data. In this unsupervised learning process, the learned network does not pay attention to the labeled target training examples, i.e, at the unsupervised meta-learning stage, the model is unaware of the task to be solved. In this paper, we target semi-supervised few-shot video learning in a more realistic scenario.

Typically, few-shot video classification models are trained on K-shot, N-way episodes [1], [4], [6]. First, each episode is constructed by sampling a subset of N classes from $\mathcal{T}_{\text{train}}$, where $\mathcal{T}_{\text{train}}$ is the meta-training set. Then, the training support set is generated as $\mathcal{S} = \{(\boldsymbol{v}_1, y_1), (\boldsymbol{v}_2, y_2), \ldots, (\boldsymbol{v}_{N\times K}, y_{N\times K})\}$ containing K examples per class. The query set $\mathcal{Q} = \{(\boldsymbol{q}_1, y'_1), (\boldsymbol{q}'_2, y'_2), \ldots, (\boldsymbol{q}_T, y'_T)\}$ are sampled from the same class, but are different samples, consisting of meta-training, meta-validation, and meta-testing splits. There are no vocabulary overlaps between the splits.

In this paper, we consider a different perspective to define the problem of incorporating unlabeled data. We leverage the unlabeled data that has been conditioned to a given task for each episode. Specifically, we denote the unlabeled dataset as $\mathcal{S}_u$. The videos from $\mathcal{S}_u$ are usually from a larger domain. The unlabeled set is also available at the meta-validation and meta-testing stage.

In this section, we introduce our framework for semi-supervised few-shot video classification. We first illustrate some preliminary components, i.e., compound memory networks. We then introduce a novel, label independent structure to store class-related information for semi-supervised learning. After that, we explain the training and evaluation protocol based on compound memory networks.

3.1 Compound Memory Networks

First, we introduce the multi-saliency embedding function that learns a fixed-size matrix representation for a variable-length video sequence. Second, we illustrate the two-layer key memory structure in compound memory network, which consists of a constituent key memory and an abstract key memory. Third, we introduce the reading and writing operations for compound memory network. Last, training loss will be explained to optimize our network.

3.1.1 Multi-saliency Embedding Function

In this section, we introduce a multi-saliency embedding function for video feature learning. The multi-saliency embedding function takes variable lengths video frames as inputs and produces a fixed-size 2D matrix representation, which explores the video temporal dynamics and detect the saliency for each frame with a hidden descriptor. This 2D matrix representation encodes a sequence of video frames which will be stored to compound memory networks. [5] leverages standard convolutional networks or sequence-to-sequence networks to encode inputs. Our multi-saliency embedding function is designed for better video feature learning, and it enables the detection of different salient parts and aggregates different lengths of videos frames to produce a fixed-size representation. We denote a video as $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{m'}\}$, where m' is the number of video frames. Each element $\mathbf{p}_i$ ($i = \{1, \ldots, m'\}$) is a frame-level representation extracted by a ConvNet. We aim to aggregate a video sequence $\mathbf{P}$ into a fixed-size 2D matrix representation $\mathbf{Q}$, where the representation $\mathbf{Q}$ consists of m stacked hidden descriptors. We denote the fixed-size representation as $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$. The size of each descriptor $\mathbf{q}_i$ ($i = \{1, \ldots, m\}$) is d_q. Note that the number of video frames (m') varies across different videos, but the number of descriptor (m) is fixed.

Our multi-saliency embedding function (`MEF`) introduces a hidden variable $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_m\}$ with m components. Each component $\mathbf{h}_j$ ($j = \{1, \ldots, m\}$) is used to detect one saliency in a video. Given a video feature $\mathbf{p}_i$, a soft weight $\mathbf{a}_{ij}$ will be calculated to measure the relevance between the input $\mathbf{p}_i$ and the component $\mathbf{h}_j$. The hidden descriptor $\mathbf{q}_j$ will be the weighted sum over the residual between $\mathbf{P}$ and $\mathbf{h}_j$. The `MEF` function can be formulated by:

$$\begin{aligned} \mathbf{a}_i &= \texttt{softmax}(\frac{\mathbf{p}_i \mathbf{H}^T}{\sqrt{d_q}}), \\ \mathbf{q}_j &= \sum_{i=1}^{m'} \mathbf{a}_{ij}(\mathbf{p}_i - \mathbf{h}_j), \end{aligned} \tag{1}$$

where $\texttt{softmax}$ is defined as, $\texttt{softmax}(\mathbf{e}) = \frac{\exp(e_i)}{\sum_i \exp(e_i)}$. Following the scaled dot-product attention in [39], we use a dot-product operation to calculate the relevance score between $\mathbf{p}_i$ and $\mathbf{h}_j$. The relevance score is scaled by $\frac{1}{\sqrt{d_q}}$ before the $\texttt{softmax}$ function. In this way, the multi-saliency descriptor $\mathbf{Q}$ can be obtained from the original video sequence $\mathbf{P}$. We denote this process as $\mathbf{Q} = \texttt{MEF}(\mathbf{P}, \mathbf{H})$.

Discussion: Multi-hops attention [39], [40] shares the similar idea by calculating multiple weighted sums over the inputs. Our multi-saliency embedding function introduces an extra hidden variable $\mathbf{H}$ to enable the detection of different salient parts in videos and learn the relation between the input and hidden descriptors. The multi-saliency embedding function aggregates different lengths of videos frames and produces a fixed-size representation for subsequent operations.

3.1.2 Two-layer Key Memory

In this section, we introduce our compound memory network with the novel two-layer key memory structure. Our compound memory network is one of the Key-Value Memory Networks [5], consisting of a key memory ($\mathcal{K}$) and a value memory ($\mathcal{V}$). In [5], the key memory only stores compact vectors, while our constituent key memory and abstract key memory provide a hierarchical memory structure for the modeling of complex video dynamics. In compound memory network, we store visual information in the key memory, and the value memory saves the label information. We introduce a two-layer key memory to store 2D video representations, where the first layer is the constituent key memory ($\mathcal{C}$) and the second layer is the abstract key memory ($\mathcal{A}$). In addition, we leverage an age memory ($\mathcal{U}$) to track the usage of each memory slot. Thus, the compound memory module ($\mathcal{M}$) can be represented by the following tuple,

$$\mathcal{M} = ((\mathcal{C}_{\texttt{ns}\times\texttt{nc}\times\texttt{cs}}, \mathcal{A}_{\texttt{ns}\times\texttt{as}}), \mathcal{V}_{\texttt{ns}\times 1}, \mathcal{U}_{\texttt{ns}\times 1}), \quad (2)$$

where $\texttt{ns}$ is the memory size, $\texttt{nc}$ is the number of constituent keys, $\texttt{cs}$ is the key size and $\texttt{as}$ is the abstract key memory size.

Constituent Key Memory. We use multiple stacked constituent vectors to represent a video in constituent key memory, which provide stronger representation capabilities than a single vector representation and enable the modeling of complex video dynamics. In each constituent key memory slot, we use a multi-saliency descriptor to represent a video. Given a video $\mathbf{P}$ to be stored in constituent key memory, we can obtain the multi-saliency embedding $\mathbf{Q}$ with shape (m, d_q). We set the number of saliency descriptors (m) to be equal to the size of constituent key ($\texttt{nc}$). In this way, we directly save the video representation $\mathbf{Q}$ in the constituent key memory.

Abstract Key Memory. It offers strong representation capabilities using constituent keys. However, the introduction of constituent keys hinders the fast retrieval process during memory reading. To enable a rapid memory reading, we propose an abstract key memory upon the constituent key memory. The abstract key memory compresses the representations cached in the constituent key memory. The abstract key memory can be seen as a snapshot of the constituent key memory. The two memory modules contain the same number of slots, but they represent information at different levels. The abstract key memory contains more semantic and abstract visual features, while representations in the constituent key memory are more informative and finer.

The representation in the abstract key memory can be obtained as follows. We follow [21], [41], [42] to normalize the matrix and produce a global video representation. [41] proposed to first perform intra-normalization that ℓ_2 normalizes the sum of residuals within a coarse cluster independently, and second, ℓ_2 normalization is used to normalize the flattened global vector. Given a stacked matrix representation $\mathbf{c}_i$ ($i \in \{1, \ldots, \texttt{nc}\}$) in each constituent key memory slot, we first normalize each constituent key with ℓ_2 normalization, *i.e.*, $\|\mathbf{c}_i\| = 1$. We obtain a new matrix representation $\mathbf{C}'$ where each component has been normalized. Second, the normalized matrix representation $\mathbf{C}'$ is flattened to a vector by concatenating all components. This vector is then projected to lower dimension space by a Fully-Connected (FC) layer, producing a more compact global vector d'. Third, the global video representation is ℓ_2-normalized before storing to the abstract key memory. We denote the procedure as the $\texttt{normalize}$ function,

$$\begin{aligned} \mathbf{c}_i{}' &= \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}, \\ \mathbf{d}' &= \texttt{FC}(\texttt{flatten}(\mathbf{C}')), \\ \mathbf{d} &= \frac{\mathbf{d}'}{\|\mathbf{d}'\|}, \end{aligned} \quad (3)$$

where a $\texttt{FC}$ layer is simply a linear transformation layer, *i.e.*, $\texttt{FC}(\mathbf{x}) = \mathbf{w}\mathbf{x}+b$, and b is the bias. The compressed representation $\mathbf{d}$ will be saved to the corresponding abstract key memory. The representations in the abstract key memory will be dynamically updated when the corresponding constituent key is altered during memory updates. Each abstract key memory slot retains a one-to-one mapping to the constituent key memory slot. Our abstract key memory will accelerate the memory reading process, while the representation capability is maintained in the constituent key memory.

3.1.3 Reading and Writing Operations

Reading. Given a query vector $\mathbf{z} = \texttt{normalize}(\mathbf{Q})$, we aim to retrieve the relevant memory slots by a nearest neighbour search over the abstract key memory $\mathcal{A}$. We use cosine similarity to measure the distance between the query and the representations in abstract key memory,

$$sim(\mathbf{z}, \mathcal{A}[i]) = \frac{\mathbf{z} \cdot \mathcal{A}[i]}{\|\mathbf{z}\| \cdot \|\mathcal{A}[i]\|}. \quad (4)$$

Cosine similarity has been widely used in few-shot learning, which have been found to be generalizable in many methods [1], [5], [43]. We select the memory slots that are close to the query $\mathbf{z}$ by,

$$\texttt{NN}(\mathbf{z},\ \mathcal{A}) = \text{argmax}_i(sim(\mathbf{z}, \mathcal{A}[i])). \quad (5)$$

The k-nearest slots ordered by decreasing similarity are returned by,

$$(n_1, \ldots, n_k) = \texttt{NN}_k(\mathbf{z},\ \mathcal{A}), \quad (6)$$

where n_1 is the memory slot that is most similar to the query. At the inference phase, $\mathcal{V}[n_1]$ will be our prediction for query $\mathbf{z}$.

Writing. We introduce the writing operation when new information is to be incorporated in memory. We follow [5] to update the memory, and we extend the memory update operations by introducing the operations to jointly modify constituent key memory and abstract key memory. The abstract key memory will be dynamically updated when the corresponding constituent key is altered during memory updates. The new information reflects the relation of a new query $\mathbf{Q} = \{\mathbf{q_1}, \ldots, \mathbf{q_m}\}$ and its corresponding label y. $\mathbf{Q}$ will be written to the constituent key memory, and y will be updated in the value memory. We do not modify the memory via backpropagation, but update the memory with the following rule. First, we locate the memory slot index to be updated by performing a reading operation over the abstract key memory, which returns n_1 as the index of the nearest memory slot. Second, we replace the located memory with the new query information using two strategies as follows.

When the memory returns the correct label, *i.e.*, $\mathcal{V}[n_1] = y$, we only update the n_1 memory slot:

$$\begin{aligned} \mathcal{C}[n_1][i] &\leftarrow \mathbf{q}_i + \mathcal{C}[n_1][i], \quad \text{for } i = 1, \ldots \mathtt{nc}, \\ \mathcal{A}[n_1] &\leftarrow \mathtt{normalize}(\mathcal{C}[n_1]), \\ \mathcal{U}[n_1] &\leftarrow 0. \end{aligned} \tag{7}$$

Hence, $\mathcal{A}[n_1]$, $\mathcal{U}[n_1]$ and $\mathcal{C}[n_1]$ will be updated, while $\mathcal{V}[n_1]$ is unchanged. The new constituent key memory is generated by averaging each constituent key in $\mathcal{C}[n_1]$ and the multi-saliency descriptors $\mathbf{Q}$. When the constituent key memory is updated, the corresponding abstract key memory slot $\mathcal{A}[n_1]$ will be altered. We update the age memory by setting $\mathcal{U}[n_1]$ to 0, which indicates that the memory slot n_1 has been recently updated.

When the memory returns a wrong label, i.e.,$\mathcal{V}[n_1] \neq y$, we will incorporate the new information by storing the $(\mathbf{Q}, y)$ pair into a memory slot to reflect the information. The oldest memory slot with the largest age value in $\mathcal{U}$ will be selected, which has not been updated for a long time. We obtain the oldest memory index n' by,

$$n' = \arg\max_i(\mathcal{U}[i] + r_i), \tag{8}$$

where r_i is a random number sampled from a uniform distribution to introduce randomness during memory slot selection. After obtaining n', the memory will be updated by,

$$\begin{aligned} \mathcal{C}[n'][i] &\leftarrow \mathbf{q}_i, \quad \text{for } i = 1, \ldots, \mathtt{nc}, \\ \mathcal{A}[n'] &\leftarrow \mathtt{normalize}(\mathcal{C}[n']), \\ \mathcal{V}[n'] &\leftarrow y, \mathcal{U}[n'] \leftarrow 0. \end{aligned} \tag{9}$$

In this case, $\mathcal{V}[n']$ is also updated with the new label y.

3.1.4 Training Loss

Kaiser et al. [5] used metric learning [44] to optimize the distances between the positive samples and the negative samples. We found this ranking loss effective in optimizing memory weights, and we follow the same loss [5] to enlarge the query similarity to the positive key and minimize the similarity to the negative key. We introduce the training loss for the optimization of learnable weights in the network. Given a query $\mathbf{z}$ and a corresponding ground-truth label y, we retrieve top-k key-value pairs on memory indices $(n_1, \ldots, n_k)$ by Eq. 6. Let `i-pos` be the smallest index that $\mathcal{V}[n_{\mathtt{i\text{-}pos}}] = y$ and `i-neg` be the smallest index that $\mathcal{V}[n_{\mathtt{i\text{-}neg}}] \neq y$. We train the query vector $\mathbf{z}$ to be more similar to $\mathcal{A}[n_{\mathtt{i\text{-}pos}}]$ than $\mathcal{A}[n_{\mathtt{i\text{-}neg}}]$ with the following ranking loss,

$$\mathcal{L}(\mathbf{z}, y, \mathcal{A}) = \max(\alpha - \mathbf{z} \cdot \mathcal{A}[n_{\mathtt{i\text{-}pos}}] + \mathbf{z} \cdot \mathcal{A}[n_{\mathtt{i\text{-}neg}}], 0). \tag{10}$$

The similarity is measured by a cosine distance which compares the relevance between two vector as Eq. 4. As $\mathbf{z}$ and vectors in $\mathcal{A}$ have been ℓ_2 normalized, we omit the ℓ_2-norm notation in Eq. 10. The similarity between the query and the positive key should be larger than the similarity between the query and the negative key by margin α. The loss will be 0 when the difference between the two distances is beyond margin α.

In each episode, We clear the memory values before any operations are conducted, which initializes all memory variables to 0. The learnable weights are shared across different episodes. At the inference phase, we fix the weights of the network, while the memory module will be updated with the support set examples.

3.2 Semi-Supervised Few-shot Video Classification

To illustrate the process of leveraging unlabeled data, we visualize the entire process in Fig. 2. We first introduce how to sample examples from the set $\mathcal{S}_u$, and how to cache the most related samples to a set of label independent memory (LIM) banks. We then illustrate how to retrieve class prototypes from the LIM, which are later utilized together with the target examples to learn few-shot representation on compound memory networks.

3.2.1 Video Embedding Functions

We use YouTube-8M as the unlabeled source to facilitate few-shot training. We do not use any label annotations from this dataset. To learn the unlabeled video representation, we use the Inception-V3 network [9] to extract the frame-level representations, which have been computed by the dataset authors. The provided features used PCA to reduce the representation size to $1,024$. Then, the video-level representations are obtained by global average pooling.

Given the training examples $\mathcal{S} = \{(\boldsymbol{v}_1, y_1), (\boldsymbol{v}_2, y_2), \ldots, (\boldsymbol{v}_{N\times K}, y_{N\times K})\}$, the goal is to leverage videos from an unlabeled set $\mathcal{S}_u$, to improve generalization of the classifier to recognize the query examples. The query examples are $\mathcal{Q} = \{(\boldsymbol{q}_1, y'_1), (\boldsymbol{q}_2, y'_2), \ldots, (\boldsymbol{q}_t, y'_T)\}$, where T is the number of test examples in each episode. For each task τ, we first embed the training video i with three different networks:

$$\begin{aligned} \boldsymbol{x}_i^{\text{incept}} &= \mathtt{MEF}(\boldsymbol{x}_{ik}^{\text{incept}}), \\ \boldsymbol{x}_i^{\text{rgb}} &= \mathtt{MEF}(\boldsymbol{x}_{ik}^{\text{rgb}}), \\ \boldsymbol{x}_i^{\text{flow}} &= \mathtt{MEF}(\boldsymbol{x}_{ik}^{\text{flow}}). \end{aligned} \tag{11}$$

$\boldsymbol{x}_{ik}^{\text{incept}}$ is the frame-level representation for k-th frame, which is extracted by the Inception-V3 network. We fix the network weights during the whole learning process. The same pre-trained weights are used as the frame-level feature extractor for YouTube-8M videos. Video-level features are generated by the MEF function. We denote the obtained video-level feature as $\boldsymbol{x}_i^{\text{incept}}$. The purpose of this network is to build

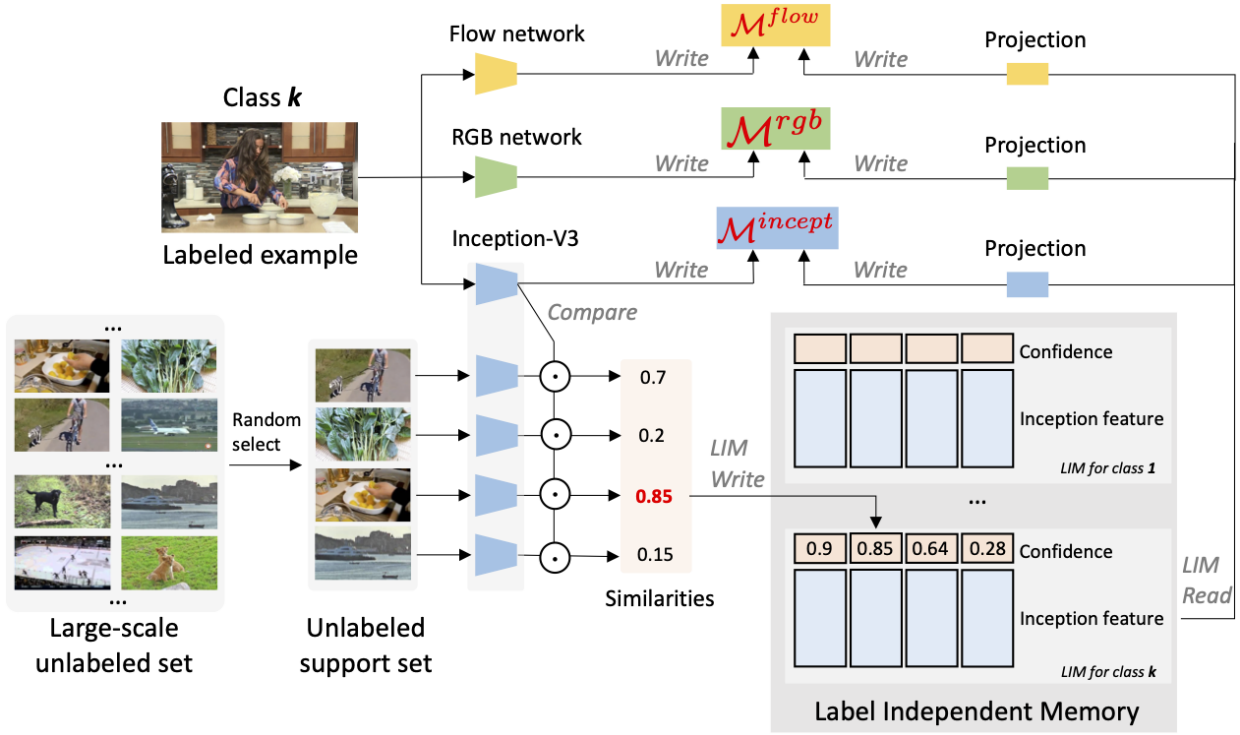

Fig. 2: Our framework for semi-supervised few-shot video classification. Given training examples and their labels, the network first finds similar instances from a large-scale unlabeled set. The examples with high similarities will be retrieved. These examples will then be cached in a label independent memory bank (LIM). This LIM has a key-value structure, where the key is the feature and the value is the confidence of this feature belonging to a specific label. After several selecting iterations, the reading operation will generate class prototype from the LIM. In this paper, three networks are used, i.e., Inception-V3, ResNet-50, and ResNet-18, which store features of Inception-V3, RGB inputs, and flow inputs, respectively. The generated class prototype will then be saved to the memory banks.

connections between the unlabeled data and the few-shot target data, which offers a fixed similarity measurement function in feature distance comparisons. This also provides a more stable training process..

$\boldsymbol{x}_{ik}^{\text{rgb}}$ is the frame-level output from the ResNet-50 network with RGB inputs. Frame-level features are also encoded by the MEF function. This network is pre-trained on ImageNet, and it is updated during the few-shot training process. The bottom layers are fixed until the res_5 block. The goal of this network is to model the objects, scenes, and the environments in the target dataset.

$\boldsymbol{x}_{ik}^{\text{flow}}$ denotes the clip-level feature for stacked clip k. They are extracted from a ResNet-18 network with optical flow as inputs. Stacked optical flows are fed to the network as input "images". In the absence of an effective pre-trained optical flow model to initialize data from, we choose to use ResNet-18 as it is relatively shallow and easy to optimize. The optical features are also be encoded by the MEF function.

For an unlabeled video u_j in $\mathcal{S}_u$, we extract the video-level representation by,

$$\boldsymbol{u}_j^{\text{incept}} = \text{MEF}(\boldsymbol{u}_{jk}^{\text{incept}}). \tag{12}$$

Thus, $\boldsymbol{x}_i^{\text{incept}}$ and $\boldsymbol{u}_j^{\text{incept}}$ are parameterized by the same Inception-V3 network. We fix all the parameters for feature extraction of unlabeled data, which bypasses additional learning of its representation. If we train the feature extraction network on set $\mathcal{S}_u$, it could introduce more difficulties when modeling the data distribution on a large dataset. The labels are unknown on the large unlabeled dataset. Thus, we choose to fix the feature extraction network for $\mathcal{S}_u$ to ease the learning difficulties.

To summarize, we now obtain the video-level representations for both unlabeled set and the training set. For an unlabeled video, a video-level representation is used to compare it with the target training examples. In addition, the target training examples are encoded with a ResNet-50 and a ResNet-18 network, which are optimized during the training process to better represent the target dataset.

3.2.2 Unlabeled Data Selection

To leverage the unlabeled data for few-shot classification, one of the possible solutions is to assign a pseudo label to the unlabeled data. However, assigning a specific label to each data point can possibly introduce noises to the classifier. In [2], the prototypical network is proposed to learn to compare among prototype representations of each class. Our proposed label independent memory cache is different from [36] and [2]. We first select the related unlabeled examples by comparing the distances between the query and the unlabeled data. Specifically, we randomly sample n_u examples from the set $\mathcal{S}_u$. We denote the features of the sampled videos as $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{n_u}$. These examples can be regarded as an unlabeled "support" set to the training examples. Each training pair can be denoted as ($\boldsymbol{x}^{\text{incept}}$, $\boldsymbol{x}^{\text{rgb}}$,

Algorithm 1 The pipeline of our semi-supervised few-shot video classification.

Input:
v, y: training video and its label
q, y': query video and its label
$\mathcal{U}$: Unlabeled video set
n_{all}: the number of videos to be sampled in $\mathcal{S}_u$
$\boldsymbol{I}_k$: Label independent memory for class k
$\boldsymbol{M}^{\text{incept}}$, $\boldsymbol{M}^{\text{rgb}}$, $\boldsymbol{M}^{\text{flow}}$: Compound Memory Networks
Output: The loss $\mathcal{L}_{\text{all}}$ for each iteration.
$\boldsymbol{x}^{\text{incept}}$: MEF encoded Inception-V3 feature for video v
$\boldsymbol{x}^{\text{rgb}}$: MEF encoded ResNet-50 feature for video v
$\boldsymbol{x}^{\text{flow}}$: MEF encoded ResNet-18 feature for video v
while total sampled videos $\leq n_{all}$ **do**
$u_1, u_2, \ldots, u_{n_u} \leftarrow$ sample n_u videos from $\mathcal{S}_u$
$\boldsymbol{u}_1^{\text{incept}}, \boldsymbol{u}_2^{\text{incept}}, \ldots, \boldsymbol{u}_{n_u}^{\text{incept}} \leftarrow$ MEF encoded Inception-V3 feature for each video
$\boldsymbol{u}_j$, $\alpha_j \leftarrow$ NN($\boldsymbol{x}^{\text{incept}}$, $\boldsymbol{u}_*^{\text{incept}}$) {return top-$k$ examples that are most similar to $\boldsymbol{x}^{\text{incept}}$}
Write $\boldsymbol{u}_j$, α_j to $\boldsymbol{I}_y$
end while
Read $\boldsymbol{e}^{\text{incept}}$, $\boldsymbol{e}^{\text{rgb}}$, $\boldsymbol{e}^{\text{flow}}$ from $\boldsymbol{I}_{y'}$
Using ($\boldsymbol{e}^{\text{incept}}$, $\boldsymbol{x}^{\text{incept}}$), ($\boldsymbol{e}^{\text{rgb}}$, $\boldsymbol{x}^{\text{rgb}}$), ($\boldsymbol{e}^{\text{flow}}$, $\boldsymbol{x}^{\text{flow}}$) to update $\mathcal{M}^{\text{incept}}$, $\mathcal{M}^{\text{rgb}}$, $\mathcal{M}^{\text{flow}}$, respectively
Calculate $\mathcal{L}_{\text{all}}$ using Equation 17

$\boldsymbol{x}^{\text{flow}}$, y). It first performs nearest neighbour search on the sampled unlabeled examples using $\boldsymbol{x}^{\text{incept}}$,

$$\alpha_j = \frac{\boldsymbol{x}^{\text{incept}} \cdot \boldsymbol{u}_j}{||\boldsymbol{x}^{\text{incept}}|| \cdot ||\boldsymbol{u}_j||}, \quad NN(\boldsymbol{x}^{\text{incept}}, \boldsymbol{u}) = \text{argmax}_j \alpha_j, \tag{13}$$

where cosine similarity is used to measure the relevance between the labeled query and the unlabeled examples.

After a nearest neighbour search, the top-k features and their similarities to the query $\boldsymbol{x}^{\text{incept}}$ will be returned. The similarity α_j will be regarded as a confidence score that indicates $\boldsymbol{u}_j$ belonging to label y. A larger α_j means the probability of $\boldsymbol{u}_j$ belonging to y is higher. Note that conducting a nearest neighbour search on a large number of unlabeled examples might lead to the out of memory error. We divide the unlabeled data into batches for the query to compare each batch individually, where the most similar unlabeled instances will be cached in the label independent memory. To be specific, the label independent memory will store the prediction pair ($\boldsymbol{u}_j$, α_j, y).

3.2.3 Label Independent Memory

The generated pair ($\boldsymbol{u}_j$, α_j, y) will be cached in the LIM bank. For a K-shot task, K LIM banks will be used to store examples for K different classes. Each class k has a corresponding memory bank $\boldsymbol{I}_k$. Each memory bank has its key and value, where the keys are the feature representation $\boldsymbol{u}$ from the unlabeled dataset, and the value is the confidence α of the feature belonging to class y. Usually, a reading and a writing operation are applied to a typical memory network. These operations are explained as follows.

Writing. In the writing operation, we first locate the y-th memory bank where the ($\boldsymbol{u}_j$, α_j, y) pair needs to be written to. We denote the target memory bank as $\boldsymbol{I}_y$. Each $\boldsymbol{I}$ has a fixed number of slots n_c, and a position variable p is used to record the number of valid memory slots that have been used. The memory slots are sorted using the value part of the memory, where the higher values are ordered first, followed by lower values. The new ($\boldsymbol{u}_j$, α_j) pairs will be inserted to the memory to keep the memory ordered. Specifically, the new feature will be written to position p via

$$\boldsymbol{I}_p^k \leftarrow \boldsymbol{u}_j, \quad \boldsymbol{I}_p^v \leftarrow \alpha_j, \tag{14}$$

where $\boldsymbol{I}_p^k$ is the key memory slot to save the features at position p, and $\boldsymbol{I}_p^v$ is the value memory slot to save the confidence score at position p. After the write operation, p is updated by $p = max(p+1, n_c)$.

Reading. We introduce the reading operation of the memory. Each class has a corresponding memory bank. The memory bank stores visual examples that can be searched by the queries in the same class. These examples can be used as labeled noisy data for few-shot training. The reading operation is applied on each memory bank, and the fetched data becomes the prototype for the corresponding category. The prototype for each class is calculated by the weighted average of the features in the valid slots,

$$\boldsymbol{e}_c^{\text{incept}} = \frac{\sum_{i=1}^{p} \boldsymbol{I}_i^v \cdot \boldsymbol{I}_i^k}{\sum_{i=1}^{p} \boldsymbol{I}_i^v}, \tag{15}$$

where $\boldsymbol{e}_c^{\text{incept}}$ will be the prototype for class c.

To generate the prototype of class c for different networks, i.e., ResNet-50 and ResNet-18, we add another projection network which attempts to map the Inception-V3 feature to the other two features. Specifically, we use a three-layer multilayer perceptron (MLP), which consists of the structue of FC-ReLU-Dropout-FC-ReLU-Dropout-FC. The prototype of class c is represented by,

$$\boldsymbol{e}_c^{\text{rgb}} = \frac{\sum_{i=1}^{p} \boldsymbol{I}_i^v \cdot \text{MLP}_{\text{rgb}}(\boldsymbol{I}_i^k)}{\sum_{i=1}^{p} \boldsymbol{I}_i^v}, \quad \boldsymbol{e}_c^{\text{flow}} = \frac{\sum_{i=1}^{p} \boldsymbol{I}_i^v \cdot \text{MLP}_{\text{flow}}(\boldsymbol{I}_i^k)}{\sum_{i=1}^{p} \boldsymbol{I}_i^v}. \tag{16}$$

Thus, three different prototypes are obtained for each class c, which are $\boldsymbol{e}_c^{\text{incept}}$, $\boldsymbol{e}_c^{\text{rgb}}$, and $\boldsymbol{e}_c^{\text{flow}}$.

Training. In semi-supervised training, we have two inputs: one is the labeled support examples and the other is the class prototype learned from the labeled independent memory bank. We have three types of networks, and we propose three CMNs, denoted as $\mathcal{M}^{\text{incept}}$, $\mathcal{M}^{\text{rgb}}$, and $\mathcal{M}^{\text{flow}}$, respectively. The inputs to each CMN are ($\boldsymbol{e}_y^{\text{incept}}$, $\boldsymbol{x}_i^{\text{incept}}$), ($\boldsymbol{e}_y^{\text{rgb}}$, $\boldsymbol{x}_i^{\text{rgb}}$), and ($\boldsymbol{e}_y^{\text{flow}}$, $\boldsymbol{x}_i^{\text{flow}}$). The memory will be updated independently, using both $\boldsymbol{x}_i^*$ and $\boldsymbol{e}_y^*$. Once updated, features from the unlabeled data set become more similar to features in the labeled target datasets.

The final training ranking loss is formulated by,

$$\mathcal{L}_{\text{all}} = \mathcal{L}(\boldsymbol{z}^{\text{incept}}, y, \mathcal{M}^{\text{incept}}) + \mathcal{L}(\boldsymbol{z}^{\text{rgb}}, y, \mathcal{M}^{\text{rgb}}) + \mathcal{L}(\boldsymbol{z}^{\text{flow}}, y, \mathcal{M}^{\text{flow}}), \tag{17}$$

where $\mathcal{L}$ is defined in Eq. 10, $\boldsymbol{z}$ is the query and y is the label.

At the test stage, we evaluate two cases of unlabeled data. In the first case, the unlabeled data are available, while

in the second case, the unlabeled data are inaccessible. We expect the more efficient embedding networks of ResNet-50 and ResNet-18 will be learned when training with large amount of unlabeled data. We illustrate the training details of our whole framework in Algorithm 1.

4 EXPERIMENTS

4.1 Datasets

We collected two datasets for few-shot video classification evaluation, which have been released for future research. In our collected "Kinetics-100", we used videos from the recently released Kinetics dataset [45], which consists of 400 categories and 306,245 videos, covering videos from a wide range of actions and events, e.g., "dribbling basketball", "robot dancing", "shaking hands", and "playing violin". We randomly selected 100 classes from the Kinetics dataset, each of which contains 100 examples. We additionally collected "Something-Something-100" [31] on Something-Something V2 [46]. Similar to Kinetics-100, we selected 100 classes from Something-Something V2, where each category has 100 examples. The 100 classes were split into 64, 12 and 24 non-overlapping classes to use as the meta-training set, meta-validation set and meta-testing set, respectively. The splits can be found in https://github.com/ffmpbgrnn/CMN.

We used YouTube-8M v1 as the source of unlabeled video data. The YouTube-8M [8] dataset is imbalanced, with some categories having over 50K positive examples, while other categories have only 100 positive examples. YouTube-8M consists of around 8 million videos, with total length of 500K hours. The average length of the videos in the dataset is 230 seconds. This dataset provides profound visual and motion dynamics that can be readily used.

4.2 Implementation Details

In an n-way, k-shot problem, we randomly sampled n classes. Each class has k examples, while an additional unlabeled example belonging to one of the n classes is used for testing. Thus each episode has $nk + 1$ examples. We calculated the mean accuracy by randomly sampling 20,000 episodes in all experiments.

For ResNet-50 and ResNet-18, we followed the standard image preprocessing procedure in [47], [48], whereby the image was first rescaled by resizing the short side to 256 and a 224×224 region was randomly cropped from the image. We cropped the central region during the inference phase. For Inception-V3, we resize the input images to 299×299 before forwarding the inputs to the network [9].

We kept the default network parameters for training. The weight decay is set to 1×10^{-4}. For training ResNet-18, we take 10 stacked optical flow images as inputs to the network. ResNet-18 is also pre-trained on ImageNet.

We optimized our model with Adam [49] and fixed the learning rate to 1.0×10^{-4}. The margin α was set to 0.5 in all experiments. During memory slot selection, the random variable r_i is introduced to improve model robustness. r_i is sampled from a uniform distribution in the range [-8.0, 8.0). We tuned the hyper-parameters on the meta-validation set, and stopped the training process when the accuracy on the meta-validation set began to decrease. The model was implemented with the TensorFlow framework [50].

The LIM memory size was set to 512. The batch size was 16. At each iteration of unlabeled video sampling, we use 500 videos. Thus, ten iterations are needed to iterate over 5,000 videos. The memory size for CMN is 1,024. The model is implemented by TensorFlow [50].

4.3 Evaluation on Few-shot Video Classification

We first present the results on supervised few-shot video classification, where no external data are utilized.

4.3.1 Comparisons to Baselines

We compare our model with several baselines. We report 1-shot, 2-shot, 3-shot, 4-shot and 5-shot results on the 5-way classification task. In the first baseline, we utilize all training data to pre-train the ResNet-50 network. At the testing stage, we fine-tune the network for each episode. The network is initialized with the pre-trained weights up to the last layer. The weights in the last layer is randomly initialized. We test the performance with different inputs. For "RGB w/o mem", we use RGB frames as inputs to train the network. For "Flow w/o mem", stack flows images are stacked as inputs to the network. To encode videos with a more sophisticated embedding function upon the frame-level features, we use an LSTM to aggregate temporal dynamics in each video. The LSTM takes the RGB features as inputs and is fine-tuned for each episode. We denote this baseline as "LSTM (RGB) w/o mem". Another baseline is a nearest neighbour baseline ("Nearest-finetune"). First, we finetune the ResNet-50 network to classify all classes in the meta-training set. Next, we feed each frame as the input image, and the video-level label is used as the label for each frame. We initialize the weights of the ResNet-50 network with the ImageNet pre-trained model. We train the network via stochastic gradient descent (SGD) with momentum 0.9. We set the initial learning rate to 0.01. We decrease the learning rate by 0.1 every 10 epochs. The batch size is 128. During inference, we feed the video frames to the finetuned ResNet-50 network and extract the activations from the last layer before final classification. We average the frame-level features and obtain a video-level representation with a dimension of 2,048 dimension. Furthermore, we apply a ℓ_2 normalization before nearest neighbour search.

In the next baseline ("Nearest-pretrain"), we do not fine-tune the ResNet-50 network on the meta-training dataset, but directly utilize the pre-trained weights without modification. We embed the video with the same procedure in "Nearest-finetune", and then apply nearest neighbour search.

We also show the result of the Matching Network [1] ("MatchingNet") on this dataset, which achieves state-of-the-art performance on the few-shot image classification task. We implement the Matching Network algorithms ourselves. First, we feed the frames to a ResNet-50 network without fine-tuning. We average frame-level features to obtain a video-level feature. Then, we use the fully-conditional embedding (FCE) function proposed in [1] to embed the training examples. The FCE uses a bidirectional-LSTM and each training example is a function of all the other examples.

TABLE 1: 5-way few-shot video classification on the meta-testing set of Kinetics-100 and Something-Something-100. The numbers are reported in percentages. Our CMN achieves best results.

	Kinetics-100					Something-Something-100				
Model	1-shot	2-shot	3-shot	4-shot	5-shot	1-shot	2-shot	3-shot	4-shot	5-shot
RGB w/o mem	28.7	36.8	42.6	46.2	48.6	20.0	25.3	29.2	30.9	33.6
Flow w/o mem	24.4	27.3	29.8	32.0	33.1	21.2	26.0	30.1	31.8	33.8
LSTM (RGB) w/o mem	28.9	37.5	43.3	47.1	49.0	19.8	24.9	28.6	30.6	32.5
Nearest-finetune	48.2	55.5	59.1	61.0	62.6	27.5	32.0	35.9	37.8	41.0
Nearest-pretrain	51.1	60.4	64.8	67.1	68.9	28.1	33.3	37.2	39.2	43.8
MatchingNet [1]	53.3	64.3	69.2	71.8	74.6	31.3	35.9	39.8	40.5	45.5
MAML [4]	54.2	65.5	70.0	72.1	75.3	30.9	35.1	38.6	40.0	41.9
Plain CMN [5]	57.3	67.5	72.5	74.7	76.0	33.4	38.9	42.5	44.0	46.5
LSTM-emb	57.6	67.9	72.8	74.8	76.2	33.0	38.5	41.8	43.8	46.2
CMN	**60.5**	**70.0**	**75.6**	**77.3**	**78.9**	**36.2**	**42.1**	**44.6**	**47.0**	**48.8**

TABLE 2: Comparisons between different memory sizes on 5-way few-shot video classification.

Model	1-shot	2-shot	3-shot	4-shot	5-shot
Mem-64	52.0	61.9	66.5	69.4	71.2
Mem-128	53.4	63.7	68.9	71.5	73.5
Mem-512	**55.1**	**65.3**	**70.1**	72.0	**74.2**
Mem-2048	55.0	65.0	69.7	**72.4**	74.1

TABLE 3: Comparisons between different numbers of multi-saliency descriptors on 5-way few-shot video classification.

Model	1-shot	2-shot	3-shot	4-shot	5-shot
Desc-1	53.7	63.5	68.3	70.9	73.3
Desc-5	**55.1**	**65.3**	**70.1**	**72.0**	**74.2**
Desc-10	53.2	62.9	68.2	70.0	72.3

To train MAML [4], we average the frame-level features and follow the default hyper-parameters in [4].

In another baseline "Plain CMN", we remove the constituent key memory from the model and use a video-level vector as video representation. We replace our embedding module with an LSTM function, while keeping the other settings the same. We denote this baseline as "LSTM-emb". We conduct this baseline to show the effectiveness of our compound memory network structure.

The results in Table 1 show our CMN improves the baselines in all shots. We observe that fine-tuning the ResNet-50 network on the meta-training set does not improve the few-shot video classification performance, but significantly harms performance. As there are no overlapping classes between the meta-training set and the meta-testing set, it is very likely that the model will overfit the meta-training set. Our CMN structure also outperforms the Matching Networks by more than 4% across all shots. Furthermore, our CMN structure outperforms the "Plain CMN", which demonstrates the strong representation capability of the constituent key memory. An improvement of about 10% is obtained between the 1-shot setting and the 2-shot setting, by only adding one example per class. The relative improvement decreases when more examples are added, e.g., the improvement from 3-shot to 4-shot is only 1.7%. This shows that one-shot classification is still a difficult problem which can be further improved in the future.

TABLE 4: Comparisons between different ways few-shot video classification.

Model	1-shot	2-shot	3-shot	4-shot	5-shot
5-way	55.0	65.0	69.7	72.4	74.1
6-way	51.7	61.8	66.4	69.3	71.2
7-way	49.5	59.6	64.3	67.1	68.9
8-way	46.0	56.1	61.0	64.0	65.8

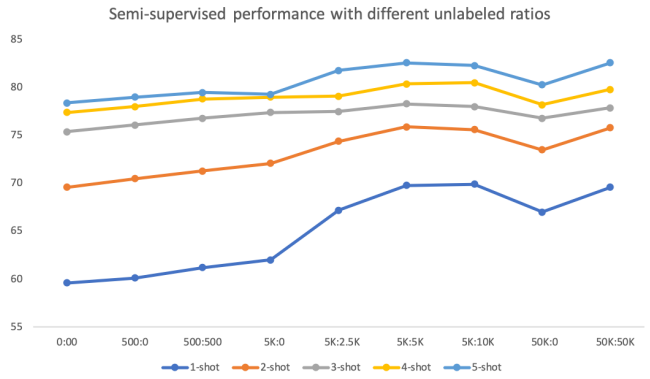

Fig. 3: Comparing different numbers of unlabeled videos for training. For example, when we use 5,000:1,000, it means 5,000 unlabeled videos are used for training, and 1,000 videos are used for inference.

4.3.2 Ablation Study

We perform ablation experiments to explain our selections for the final model on Kinetics-100. The default setting is the 5-way few-shot classification. We show the classification performance of different memory sizes in Table 2. The results of different numbers of constituent keys are shown in Table 3. We also report the results of other few-shot video classification tasks with different numbers of categories. We report the results on the meta-validation set, and choose only 10 frames during evaluation.

Memory size. The results of different memory sizes are shown in Table 2. When the memory has a small number of slots, the performance is worse because some information has to be wiped out as new data arrives. A memory size of 512 achieves the best results. Increasing the memory size does not improve performance when the memory is large enough to record all the information.

TABLE 5: 5-way semi-supervised few-shot video classification on the meta-testing set of Kinetics-100 and Something-Something-100. "Incept" denotes the features are extracted by Inception-V3.

	Kinetics-100					Something-Something-100				
Model	1-shot	2-shot	3-shot	4-shot	5-shot	1-shot	2-shot	3-shot	4-shot	5-shot
CMN w/o external data	60.5	70.0	75.6	77.3	78.9	36.2	42.1	44.6	47.0	48.8
DeepCluster CACTUs-MAML (Incept) [7]	65.1	72.8	76.5	77.9	79.5	37.9	44.5	45.9	47.8	49.9
DeepCluster CACTUs-ProtoNets (Incept) [7]	66.9	73.2	77.0	78.1	79.9	38.4	44.8	46.1	48.0	50.1
LIM-Incept (Ours)	69.8	75.9	78.3	80.4	82.6	41.1	46.9	48.0	51.5	53.0
LIM-3 modalities (Ours)	**73.3**	**78.3**	**80.8**	**82.4**	**84.0**	**44.0**	**49.8**	**51.3**	**53.9**	**55.1**

TABLE 6: Comparing our model with different modalities. We show that using optical flow information is important to learn a more power video representation.

Model	1-shot	2-shot	3-shot	4-shot	5-shot
LIM-Incept	69.8	75.9	78.3	80.4	82.6
LIM-Incept-ResNet18-Flow	72.1	77.4	79.6	81.8	83.2
LIM (full)	**73.3**	**78.3**	**80.8**	**82.4**	**84.0**

The number of multi-saliency descriptors. The results in Table 3 show that multi-saliency descriptors with stronger representation capability obtain better performance than a single descriptor. The performance decreases when too many descriptors are used, because more parameters are introduced in the network.

N-way classification. In all previous experiments, evaluations were conducted on the 5-way classification setting. n-way classification with larger n is a similar task to 5-way classification, but it can be more difficult. As can be seen in Table 4, the performance decreases when n increases.

4.4 Semi-supervised Few-shot Video Classification Results

4.4.1 Comparisons to Baselines

We compare our results on semi-supervised few-shot video classification. The results on Kinetics-100 and Something-Something-100 are shown in Table 5. We use the Inception-V3, ResNet-50 and ResNet-18 networks in this setting. The results are reported when 5,000 unlabeled videos are presented at each episode during training. The same amount of unlabeled videos are available during inference. As can be seen, it is beneficial to leverage unlabeled data for few-shot video classification. For 5-way, 1-shot video classification, LIM-Incept outperforms CMN [6] by 9.3%. For 5-way, 5-shot video classification, LIM outperforms CMN 3.7%. It shows the unlabeled data are more useful when the number of training video are quite limited. The results indicate that when there are limited training data, it is beneficial to improve the model with more unlabeled videos. Our model learns to measure the distances between the target set and the unlabeled data. We also compared our results to [7], where the unlabeled examples are first used for clustering. Note that our LIM-Incept outperforms DeepCluster, CACTUs-MAML and DeepCluster CACTUs-ProtoNets with a clear margin. We train CACTUs on YouTube-8M, and then tune it on the meta-training set. Our multiple banks implementation further improves LIM by leveraging optical flow information. In the next section, we will study several important aspects of our model, including the effectiveness of different modalities, and the number of unlabeled videos used for each episode. Our results indicate that it is effective to leverage motion cues for video classification.

We show the confusion matrix for semi-supervised few-shot learning on both Kinetics-100 (Fig. 4) and Something-Something-100 (Fig. 5). The confusion matrices show that the Something-Something-100 dataset is more challenging than Kinetics-100. For example, "Putting something upright on the table" can be easily confused with "Unfolding something", and "Scooping something up with something" can be easily confused with "Putting something on the edge of something".

4.4.2 Ablation Study

The ablation studies on semi-supervised few-shot classification is conducted on Kinetics-100.

Modality. We study the benefit of introducing multiple modalities for few-shot learning. In previous studies [6], only RGB frames are used for few-shot classification. We have shown the benefits of using three different networks in our framework. We now explicitly show that flow information is beneficial to the whole framework. The results are shown in Table 6. As can be seen, when ResNet-18 is introduced in the framework (LIM-Incept-ResNet18-Flow v.s LIM-Incept), substantial improvements are obtained for all shots experiments. Even though the optical flow information is trained using a shallow network, i.e., ResNet-18, its information is vital to the final results. Motion information has been found to be important in action recognition tasks [13]. In this paper, we first leverage motion information under the few-shot video classification setting. The results also show that motion information can be possibly be learned from only a few examples. Note that in our framework, we did not directly model motion information at the unlabeled dataset, which could be too difficult to learn. Instead, we use the Inception-V3 feature to generate the optical flow to be stored in the memory. Thus, it shows the possibly of learning a flow network that can infer flow information from static RGB cues. This can significantly

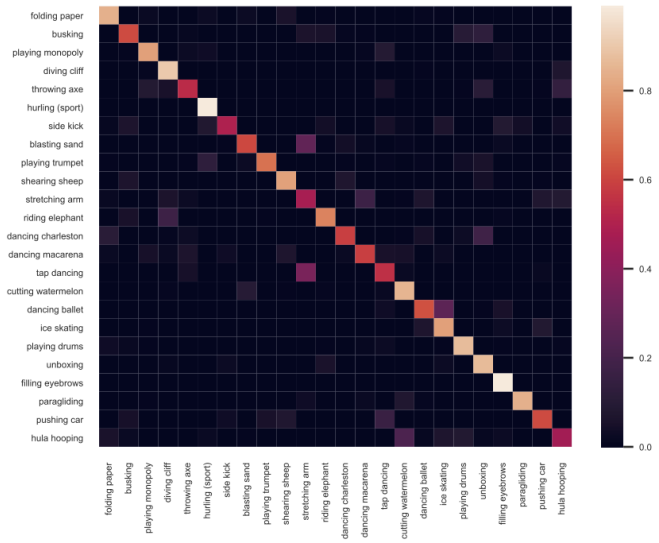

Fig. 4: Confusion matrix for semi-supervised few-shot video classification on Kinetics-100.

Fig. 5: Confusion matrix for semi-supervised few-shot video classification on Something-Something-100.

TABLE 7: Comparisons of unlabeled data from Kinetics and from YouTube-8M. We do not use any label information from both datasets. The results indicate that the similarity between the source domain and the target domain will affect the classification performance.

Model	1-shot	2-shot	3-shot	4-shot	5-shot
LIM (Kinetics)	70.5	76.8	79.0	81.2	83.3
LIM (YouTube-8M)	69.8	75.9	78.3	80.4	82.6

reduce the cost of extracting optical flow using hand-crafted algorithms.

Number of unlabeled videos. We study how the performance changes when we leverage different number of unlabeled videos during training and inference. Note that during inference stage, it is not necessary to use unlabeled data in our framework, while the support training set can also be saved to CMN for query inference. However, we would like to see how the performance changes when we have a different number of unlabeled videos during training and inference. The results are shown in Fig. 3. We compare the models under different settings. We denote the number of unlabeled videos in training as a, and the number of unlabeled videos in inference as b. We compare different $a : b$ combinations. Specifically, we evaluate $a = 500$, $a = 5K$, and $a = 50K$. As can be seen, with only a few unlabeled examples, e.g., $a = 500$, the performance is the worst. This poor performance can be due to the large variances in the sampled 500 examples, which may be totally unrelated to the target set. When no unlabeled videos are used (0:0), the task degenerates to few-shot video classification, where only labeled images are written into $\mathcal{M}^{incept}$ and no unlabeled information will be written to the memory. The results clearly show that our LIM outperforms the baseline with a large margin. We find that $a = 5K$ is a good hyper-parameter when the source dataset is YouTube-8M. When we have more unlabeled videos, the performance saturates. It shows that although the unlabeled videos are beneficial to the target data, the improvements have a limit. The source data and the targets are from different distributions, where the domain gap can also be difficult to overcome. We did not attempt $a \geq 50K$, as it is time-consuming to sample 50K examples for a single episode training.

Now we discuss the behaviours of b. As can be seen, when $b = 0$, the network can also learn to generalize. A larger b will produce a better performance, however, when we increase b further ($a : b = 5K : 10K$), the performance also saturates. This again indicates that increasing the number of unlabeled data for each episode not always helps the performance.

Different source dataset. We now demonstrate how the performance changes when we use videos from Kinetics that contain different categories to our Kinetics-100 dataset. Note that we only use the raw videos, and no label information is used. We aim to see how the source dataset will affect the performance. The results are shown in Table 7. Note that using videos from Kinetics will improve the performance. This result is not surprising, as there are similarities between our 100 classes and the remaining 300 classes. Although some categories are of different labels, there could be similarities between them. However, using YouTube-8M is a more general choice for video classification. It is common that we are not aware of the distribution of the target dataset. We believe if the targets dataset is changed to another one, YouTube-8M could possibly still benefit the few-shot training, as it contains large vocabulary and includes videos from many different domains.

4.5 Discussion

Speed Comparisons We compare the training speed and testing speed of our LIM and other baselines. We evaluate the comparison in the 1-shot setting. The results are shown in Table 8. We report the speed of both few-shot video

TABLE 8: Speed comparisons in few-shot video classification. The time is reported in hours. For the time cost of training, we report the total training time. For the testing time cost, we report the sum of 20,000 episodes.

Model	Semi	1-shot acc	Training	Testing
RGB w/o mem	✗	28.7	2.3	0.16
LSTM (RGB) w/o mem	✗	28.9	3.2	0.19
Nearest-pretrain	✗	51.1	0.0	0.08
MatchingNet [1]	✗	53.3	5.8	0.14
MAML [4]	✗	54.2	10.6	0.47
Plain CMN [5]	✗	57.3	6.3	0.17
LSTM-emb	✗	57.6	7.1	0.20
CMN	✗	60.5	8.4	0.25
DeepCluster CACTUs-MAML (Incept)	✓	65.1	17.9	0.71
DeepCluster CACTUs-ProtoNets (Incept)	✓	66.9	14.3	0.38
LIM-Incept (Ours)	✓	69.8	15.7	0.34
LIM-3 modalities (Ours)	✓	73.3	20.8	0.67

classification methods and semi-supervised few-shot video classification methods. We observe that the training time of MAML is higher than the other baselines, and our "LIM-Incept" achieves strong performance while the training cost is relatively low. "Nearest-pretrain" does not need to train a model. It only performs nearest neighbour search during testing. Note that "LIM-3 modalitites" is more expensive compared to other methods as it utilizes optical flow information. Future research studies can be conducted on improving LIM with fewer modalities but the accuracy is maintained.

Failed Attempts. In this section, we list a few methods we tried but do not work in our preliminary experiments. First, the training fails when the backbone for feature extraction of unlabeled images is not fixed. We observe that the training collapses when we tune Inception-V3 during semi-supervised few-shot learning. It might because that the Inception-V3 model is biased towards learning the video features from the large amount of unlabeled data. This might lead to collapsed few-shot training process. Second, we tried to generate pseudo labels for the unlabeled data using a specific model trained on each episode. The generated pseduo labels are then used to finetune the backbone. The pseudo label generation process is not so accurate that the subsequent finetuning performance degenerates. The "Nearest-finetune" baseline in 1-shot setting on Kinetics-100 is 48.2%. When unlabeled data are used to finetune the backbone, the accuracy becomes 41.4% which is significantly worse than the baseline. Third, we tried to incorporate the unlabeled data by a mixup operation [35]. The mixup operation mixes the unlabeled and labeled data. In 1-shot setting, we only have 5 examples per episode, while the number of unlabeled examples is 5,000. When mixup is used in 1-shot setting on Kinetics-100, the accuracy is 57.4% that is 12.4% worse than our LIM (69.8%). This method fails because the large ratio (1,000:1) between the unlabeled examples and labeled examples, making it difficult to mix the labeled data with the unlabeled data.

5 Conclusion

In this paper, we have proposed a compound memory network for few-shot video classification. This module stores matrix representations, which can be easily retrieved and updated in an efficient way. Additionally, we study the effectiveness of introducing unlabeled videos for semi-supervised few-shot video classification. We propose a novel Label Independent Memory to cache the class related examples. The returned class prototype is beneficial to learn a more stable embedding function. We also leverage multi-modal memory banks to further improve the representation power of the memory banks. Our experimental results validate that incorporating optical flow information is beneficial for few-shot video classification. This work provides more opportunities to leverage external knowledge for better generalization, even if the external data is unlabeled. For future studies, we would like to explore more efficient search algorithms that can accelerate the nearest neighbour search. We will explore more effective methods to leverage the temporal information that is unique in videos for few-shot video classification.

References

[1] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *NIPS*, 2016.

[2] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017.

[3] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *CVPR*, 2018, pp. 1199–1208.

[4] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[5] Ł. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," in *ICLR*, 2017.

[6] L. Zhu and Y. Yang, "Compound memory networks for few-shot video classification," in *ECCV*, 2018.

[7] K. Hsu, S. Levine, and C. Finn, "Unsupervised learning via meta-learning," *arXiv preprint arXiv:1810.02334*, 2018.

[8] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016.

[10] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *ICLR*, 2015.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[13] ——, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

[14] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *IJCV*, 2018.

[15] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *ICCV*, 2015.
[16] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.
[17] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*, 2016.
[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.
[19] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017.
[20] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, 2016.
[21] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," in *CVPR*, 2017.
[22] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv preprint arXiv:1706.06905*, 2017.
[23] L. Zhu, Z. Xu, and Y. Yang, "Bidirectional multirate reconstruction for temporal modeling in videos," in *CVPR*, 2017.
[24] L. Zhu and Y. Yang, "Actbert: Learning global-local video-text representations," in *CVPR*, 2020.
[25] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *CVPR*, 2018.
[26] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *ECCV*, 2018.
[27] E. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *CVPR*, 2000.
[28] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *TPAMI*, vol. 28, no. 4, pp. 594–611, 2006.
[29] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *CogSci*, 2011.
[30] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016.
[31] K. Cao, J. Ji, Z. Cao, C.-Y. Chang, and J. C. Niebles, "Few-shot video classification via temporal alignment," *arXiv preprint arXiv:1906.11415*, 2019.
[32] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.
[33] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.
[34] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *NeurIPS*, 2019.
[35] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.
[36] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," in *ICLR*, 2018.
[37] Y. Liu, J. Lee, M. Park, S. Kim, and Y. Yang, "Transductive propagation network for few-shot learning," *arXiv preprint arXiv:1805.10002*, 2018.
[38] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *NeurIPS*, 2019.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.
[40] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*, 2017.
[41] R. Arandjelovic and A. Zisserman, "All about vlad," in *CVPR*, 2013.
[42] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.
[43] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *CVPR*, 2018.
[44] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
[45] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.
[46] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag *et al.*, "The "something something" video database for learning and evaluating visual common sense." in *ICCV*, 2017.
[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[48] ——, "Identity mappings in deep residual networks," in *ECCV*, 2016.
[49] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
[50] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *OSDI*, 2016.