Image to Video Domain Adaptation Using Web Supervision

Andrew Kae
Yahoo Research
andrewkae@verizonmedia.com

Yale Song
Microsoft Research
yalesong@microsoft.com

Abstract

Training deep neural networks typically requires large amounts of labeled data which may be scarce or expensive to obtain for a particular target domain. As an alternative, we can leverage webly-supervised *data (i.e. results from a public search engine) which are relatively plentiful but may contain noisy results. In this work, we propose a novel two-stage approach to learn a video classifier using webly-supervised data. We argue that learning appearance features and temporal features sequentially, rather than jointly, is an easier optimization for this task. We show this by first learning an image model from web images, which is used to initialize and train a video model. Our model applies domain adaptation to account for potential domain shift present between the source domain (webly-supervised data) and target domain, and also accounts for noise by adding a novel attention component. We report results competitive with state-of-the-art for webly-supervised approaches (while simplifying the training process) on UCF-101 and also evaluate on Kinetics for comparison.*

1. Introduction

Action recognition in videos is a well-studied problem in computer vision with many important applications in areas such as surveillance, search, and human-computer interaction. Modern approaches to video action recognition are based on deep neural networks, which typically require a large labeled dataset. However, it may be difficult to obtain sufficient labeled data because it may be too scarce or too expensive to obtain. We can instead leverage *webly-supervised* data (i.e. results from a public search engine) which are relatively plentiful but may be noisy.

In this paper, we present a webly-supervised approach to learn video models based on noisy web images and videos. The high-level overview of our model is shown in Figure 1. The noisy web image and web video domains are considered source domains that we want to domain adapt into a target domain. We present a two-stage approach to first learn an image model using a 2D-CNN, transfer the learned spatial weights to a 3D-CNN, and continue training a video model. Since our goal is to learn a video classifier, we can potentially learn from web videos only, but we argue that our proposed two-stage process is more appropriate for learning from noisy, webly-supervised data. Web images are typically higher resolution while web videos are typically lower resolution and may contain motion blur and other artifacts. In addition, web videos may contain many frames that are irrelevant to the target concept. Thus it may be easier to learn spatial features first, based on the relatively cleaner web images, and then learn temporal features afterward. Previous work [26] has also hypothesized that it may be difficult to learn both spatial and temporal features simultaneously. We present empirical results in Section 4 showing that our two-stage approach, which separates learning appearance and temporal features, outperforms a model that learns both jointly.

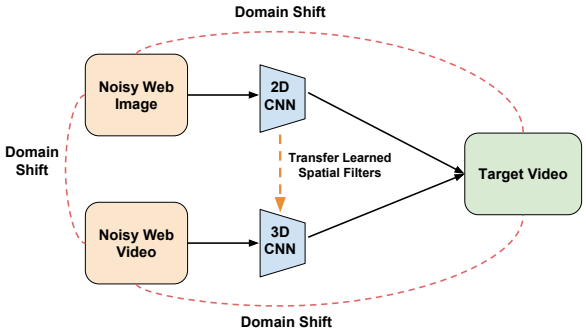

Figure 1: Given webly-supervised images and videos (source domains), we learn a video classifier for the target domain. The model is learned in a two-stage process by 1) learning an image model (2D-CNN) and 2) transferring the spatial filters to the video model (3D-CNN) to continue training. The model also accounts for domain shift and noise present in the webly-supervised data.

To account for domain shift, domain adaptation has been successfully applied for tasks such as mapping from MNIST [15] to StreetView digits [28, 10], RGB to depth images [28] and webcam to product images [10]. In our work we incorporate an adversarial training component

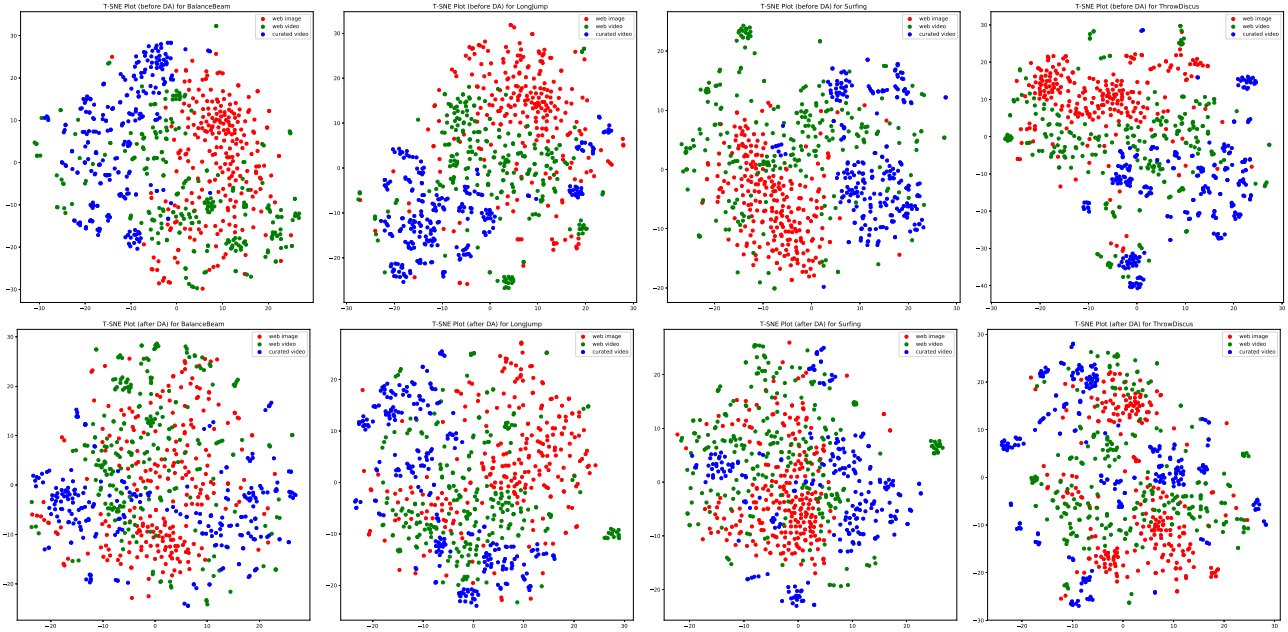

Figure 2: **Embeddings before and after domain adaptation.** We randomly sampled embeddings from the web image (**red** points), web video (**green** points) and target video (**blue** points) domains and show the T-SNE [29] plots of 4 actions: balance beam, long jump, surfing, and throw discus (from UCF-101 [24]). The top row contains the T-SNE plot before domain adaptation using pre-trained RN-34 [12] embeddings and the bottom row shows embeddings of the same actions after domain adaptation. Plot best viewed in color.

taken from Generative Adversarial Networks (GAN) [11]. To account for the noise present in webly-supervised data, we incorporate a novel attention component to reduce the effect of irrelevant examples.

In this work, the target domain consists of *curated* videos, each containing only a single concept. We consider these curated videos to be a separate domain from web images and web videos. We assume there are relatively few irrelevant chunks from videos in the target domain compared to web videos. For example, this setting may be appropriate if the target domain was surveillance videos.

To check whether there is indeed a difference between the separate domains, we extracted embeddings from random images/frames from each domain using ResNet-34 [12] and visualized T-SNE [29] plots for four different action categories from UCF-101 [24]: Balance Beam, Long Jump, Surfing, Throw Discus. The top row in Figure 2 corresponds to the embeddings before domain adaptation for curated video frames (**blue** points), web video frames (**green** points), and web images (**red** points). The bottom row corresponds to the embeddings after domain adaptation (details in Section 3). Before domain adaptation, there are visibly distinct regions corresponding to the three domains of web images, web videos and curated videos (we used UCF-101 [24] videos), which may indicate domain differences. Afterward the different domains are packed closer together, which may indicate less domain separation.

To summarize, our contributions include:

- A novel two-stage approach to first learn spatial weights from a 2D-CNN and then transfer these weights to a 3D-CNN to learn temporal weights.
- A novel attention component to account for noise present in webly-supervised data.
- Results competitive with state-of-the-art video classification on UCF101 [24] for webly-supervised approaches, while simplifying training.

2. Related Work

Webly-Supervised Learning. Previous work using webly-supervised data include [22, 4, 18, 30]. In addition, Gan *et al.* [7] jointly match images and frames in a pre-processing step before using a classifier while LeadExceed [9] uses multiple steps to filter out noisy images and frames. In contrast, our model does not have pre-processing steps and learns to downweight noisy images as part of model training. Li *et al.* [16] use web images to perform domain adaptation and learn a video classifier, but they manually filter out irrelevant web images beforehand whereas we incorporate this step into our model. DECK [8] applies pretrained image and video classifiers to web image search results in order to learn high-level concepts, but their approach is focused on zero-shot recognition.

There has also been related work in using attention for weakly-supervised learning. Zhuang *et al.* [33] stack noisy web image features together with the assumption that at least one of the images is correctly labeled. They then learn an attention model to focus on the correctly labeled images. UntrimmedNet [30] generates clip proposals from untrimmed web videos and also incorporates an attention component for focusing on the proposals with the correct action. In contrast, our model learns from both images and videos and ties attention closely with domain adaptation.

Video Classification. 3D-CNN video models such as C3D [25], P3D [20], I3D [3], R(2+1)D [26], SlowFast [6] are appealing for video classification since they learn appearance and motion features jointly. I3D [3] uses full 3D filters, while R(2+1)D [26] and P3D [20] decompose the spatio-temporal convolution into a spatial convolution followed by a temporal convolution. The design of our 3D-CNN is partly inspired by these latter approaches because of this elegant decomposition, which allows us to reuse spatial filters from a conventional 2D CNN. We could potentially use the same bootstrapping technique to inflate 2D to 3D filters as in I3D [3], but initializing and fixing the 2D filters may allow for easier training (more detail in Section 3).

Domain Adaptation. There has been much work in applying GANs [11] for domain adaptation. Models such as PixelDA [2] learn to generate realistic-looking samples from the source distribution, while others such as DANN [10] learn a domain-invariant feature representation. We adopt this latter approach in our work.

Other related works include ADDA [28] which learns a piecewise model by pre-training a classifier on the source domain and then adds the adversarial component later. Tzeng et. al [27] learn domain-invariance by incorporating a domain confusion loss (similar to a discriminator loss) and transferring class correlations between domains to preserve class-specific information. Luo *et al.* [17] propose a similar model to ours but for a fully-supervised setting, and add a semantic-transfer loss to encourage transfer of class-specific information. The main difference between our model and these approaches is that we use webly-supervised data and assume the source and target domains may contain noisy labels, which is a considerably more difficult, yet practical scenario.

Lastly, Zhang *et al.* [32] propose a similar approach to our own for image tasks, which also includes a domain-adversarial component and performs instance weighting to account for noise in the source data. However, they use a dual-discriminator approach for instance weighting whereas we use an attention-based component.

3. Model

Our goal is to learn a video classifier in the target domain by training on the (source) webly-supervised image and video domains. We propose a two-stage approach by first learning an image model using a standard 2D-CNN, transferring the learned spatial weights to a 3D-CNN and then continuing training on videos. We learn a separate model for images and videos since it may be difficult to learn appearance and motion features simultaneously.

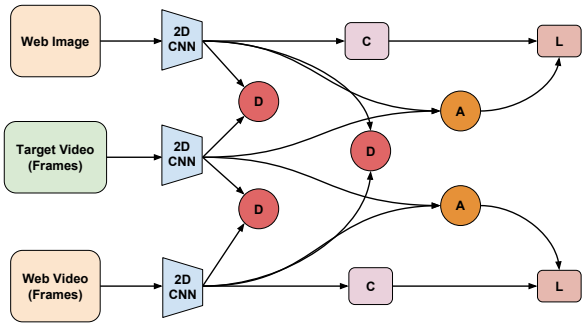

Figure 3: **Image Model**. Triplet network with branches corresponding to web images, web video frames and target video frames. We add discriminators D to enforce domain invariance between the separate domains and add attention components A to downweight irrelevant examples. C corresponds to the classifiers and L corresponds to the losses.

Our model (1) learns appearance features in the image model and motion features in the video model (2) transfers the learned spatial weights from the image model to the video model (3) accounts for noise present in the webly-supervised data and (4) performs domain adaptation from the webly-supervised domain to the target domain.

The image model shown in Figure 3 is a triplet network that performs both domain adaptation and attention-based filtering of noisy images. The three branches correspond to web images, web video frames, and target video frames (without labels). The image model learns domain invariance between the different domains and also uses an attention component to downweight irrelevant web images and web video frames, with respect to the target video frames. Intuitively the attention component downweights web images/frames that look different from target video frames.

The video model shown in Figure 4 is a Siamese network with branches corresponding to web videos and target videos (without labels). Note that the inputs are now video chunks rather than images. The spatial weights in the video model are initialized from the image model spatial weights and fixed (as indicated by the dashed lines in Figures 4 and 5). Similar to the image model, the video model also contains domain adaptation and attention components.

3.1. Notation

Let us define following notation:

- E: encoder (either a 2D or 3D CNN) returns $\mathbb{R}^d$
- C: classifier returns predictions among L labels

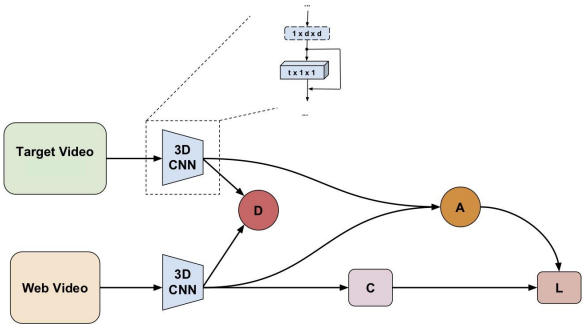

Figure 4: **Video Model**. Siamese model with branches corresponding to web video and curated video chunks. We initialize the spatial weights in the 3D-CNN from the image model (Figure 3) and add an attention component A to reduce the noise from irrelevant shots or incorrect labels. C corresponds to the classifier and L corresponds to the loss.

- N^I, N^V, N^T: number of webly-supervised image and videos, and curated (target) videos respectively
- $\mathbf{X}^I = \{x_i^I, y_i^I\}_{i=1}^{N^I}$: set of webly-supervised images where x_i^I is the ith image and y_i^I is its corresponding web label where $y_i^I \in \{1..L\}$
- $\mathbf{X}^V = \{x_j^V, y_j^V\}_{j=1}^{N^V}$: set of webly-supervised videos where x_j^V is the jth video and y_j^V is its corresponding web label and $y_j^V \in \{1..L\}$. Each video x_j^V consists of frames $\{x_{jf}^V\}_{f=1}^{N_j^V}$ where N_j^V is the number of frames in video x_j^V
- $\mathbf{X}^T = \{x_k^T\}_{k=1}^{N^T}$: set of curated videos where x_k^T is the kth video. Each video x_k^T consists of frames $\{x_{kf}^T\}_{f=1}^{N_k^T}$ where N_k^T is the number of frames in video x_k^T

3.2. Classification

We use ResNet [12] as the base architecture for both our image and video models, along with the standard softmax cross-entropy loss to train a classifier for both web images and web video frames. The losses are computed as

$$L_{wimage} = \mathbb{E}_{x^I}\big[-y^I \cdot \log(C(E(x^I)))\big]$$
$$L_{wframe} = \mathbb{E}_{x^V}\big[-y^V \cdot \log(C(E(x^V)))\big]$$

where the expectations are taken over examples x^I and x^V and y^I, y^V are their corresponding webly-supervised labels.

3.3. Domain Adaptation

We learn an encoder E that can produce feature embeddings that are indistinguishable between different domains in an adversarial setting [11]. The discriminator D tries to distinguish between embeddings generated from different domains (shown in Figure 3). By optimizing over a min-max objective, E learns embeddings that can eventually "fool" D, thus learning a domain-invariant feature representation. We define our domain-adaptation loss as

$$L^I = \mathbb{E}_{x^T}\big[\log D(E(x^T))\big] + \mathbb{E}_{x^I}\big[\log(1 - D(E(x^I)))\big]$$
$$L^V = \mathbb{E}_{x^T}\big[\log D(E(x^T))\big] + \mathbb{E}_{x^V}\big[\log(1 - D(E(x^V)))\big]$$
$$L^B = \mathbb{E}_{x^I}\big[\log D(E(x^I))\big] + \mathbb{E}_{x^V}\big[\log(1 - D(E(x^V)))\big]$$
$$L_{domain} = L^I + L^V + L^B \quad (1)$$

L^I distinguishes between web images and target frames, L^V distinguishes between web video frames and target frames, and L^B distinguishes between web images and web video frames. In each term, the first component corresponds to correctly distinguishing between different domains, and the second component tries to "fool" the discriminator D. In addition, we use a multi-layer discriminator D,

$$d_l = D_l(\sigma(d_{l-1} \oplus E_l(x)))$$

where D_l is the discriminator at the l-th layer, d_l is the discriminator output at the l-th layer, $\oplus$ denotes concatenation, $E_l(x)$ is the CNN embedding from the l-th layer, and σ is the (ReLU) activation function. Intuitively we take the encoder outputs from multiple layers, concatenate them and feed them into a discriminator (a binary classifier). Similar to [17], we have empirically found this multi-layer discriminator to perform better than the single-layer version.

3.4. Attention

Learning from webly-supervised data is difficult because it is inherently noisy. For example, if we query for a term such as "archery", we may get some results containing the action of shooting a bow and arrow but we may also get advertisements for a sporting goods store, or product shots of archery equipment, which are likely less relevant for learning to recognize the action itself.

We present a novel approach for filtering noisy data inspired by work from machine translation [1]. The attention component learns to "filter out" or downweight irrelevant images/frames by comparing the images and frames in each source domain batch to the images from the target domain batch. Intuitively, images from the source domain batch that look very different to images in the target domain batch should be given low weight. For example, it is unlikely that an advertisement or product shot is going to look like frames from the target video which we assume is curated and contains only the action. In this way, we jointly learn the relevance of both web images and web videos by comparison to the target videos.

Note that this weighting is similar to the loss update from [32] but that is based on scores from a discriminator whereas our approach is based on learning a similarity function between the different domains. Unlike previous

work [7] that performs pre-processing to filter out irrelevant images/frames, our approach learns a model of relevance jointly with other components during training. In our approach we do not need to perform manual filtering or pre-processing and instead the filtering happens jointly with model training. Zhuang et.al [33] learns attention from stacking web image activations together. In contrast our model learns attention through a comparison of the source and target domain, which may provide a more direct signal for inferring the attention weights.

More formally, given a set of web images and their corresponding labels $\mathbf{X}^I = \{x_i^I, y_i^I\}_{i=1}^{N^I}$, we compute an attention score α_i for each image x_i^I such that $\sum_i^{N^I} \alpha_i = 1$ (note that we compute these attention weights per batch during training). Let $E(x^I) \in \mathbb{R}^D$ denote the CNN embedding for a given image x^I. We compute attention scores as follows

$$\begin{aligned} e_{ik} &= A(E(x_i^I), E(x_k^T)) \\ &= E(x_i^I) \cdot W \cdot E(x_k^T)^\top \end{aligned}$$

where e_{ik} indicates the similarity between web image x_i^I and target image x_k^T and A is the attention model. W is a matrix with dimension $\mathbb{R}^{D \times D}$ and parameterizes the similarity between the embeddings from different domains. The parameters for W are learned along with the rest of the model parameters. We then compute

$$m_{it} = TopT(e_{i,1:N^T})$$

where m_{it} consists of the top T scores along the ith row. In practice we observed better performance when summing over the top T scores instead of all scores in the row.

$$s_i = \sum_{t=1}^{T} m_{it}$$

We then compute the image attention weights as

$$\alpha_i^I = \frac{\exp(s_i/\tau)}{\sum_{j'=1}^{N} \exp(s_{i'}/\tau)}$$

where τ is a temperature term. α_i^I is then used to weight the image x_i^I in the cross-entropy loss. The attention weights for video frames α_j^V are computed in the same way by comparing to the target video frames.

3.5. Image Model

The image model loss can be rewritten as:

$$\begin{aligned} L'_{wimage} &= \mathbb{E}_{x^I}\big[-\alpha^I \cdot y^I \cdot \log(C(E(x^I)))\big] \\ L'_{wframe} &= \mathbb{E}_{x^V}\big[-\alpha^V \cdot y^V \cdot \log(C(E(x^V)))\big] \\ L_{image} &= L'_{wimage} + L'_{wframe} \end{aligned} \tag{2}$$

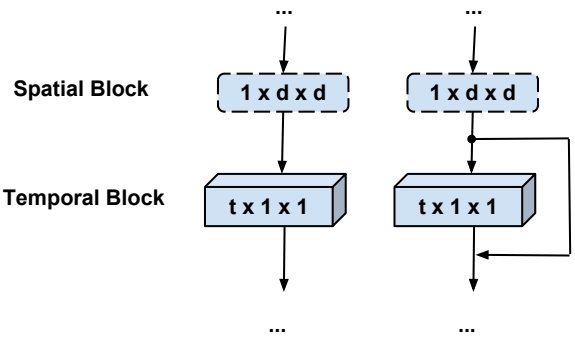

Figure 5: **Spatio-temporal Block.** (a) the decomposition of the spatiotemporal block into a 2D spatial filter followed by a 1D temporal filter (corresponds to R(2+1)D [26] model) (b) our modified block with an added residual connection. The spatial weights are initialized from the 2D CNN weights and fixed, as indicated by the dashed lines.

where the α^I is used to weight the batch. We also incorporate the domain adaptation loss from Equation 1 to get a combined loss of

$$\min_{\theta_E, \theta_C} \max_{\theta_D} L_{image}(E, C) + \beta L_{domain}(E, D) \tag{3}$$

where β is a tradeoff parameter between the weighted classification and domain adaptation terms. In practice we use the *gradient reversal layer* [10] which multiplies the gradient from the discriminator by a negative constant during backpropagation, allowing us to perform optimization in one step instead of the usual two-step optimization.

3.6. Video Model

The next step is to transfer the spatial filters learned from the image model to the video model. We assume the spatial filters have been learned appropriately from the images and we want the video model to focus on learning the motion filters. One natural way to capture this intuition is by *sequentially* arranging the spatial filter followed by the temporal/motion filter, as shown in Figure 5a. In this way we elegantly decompose the spatiotemporal kernel into a spatial filter followed by a temporal filter. Note that this formulation corresponds to the R(2+1)D [26] architecture.

Unfortunately, there is a problem with simply placing the temporal block directly after the spatial block as shown in Figure 5a for our case. The "good" spatial filters (initialized from the image model) are now interleaved with *untrained* temporal filters, which means it is possible the output distribution of the spatiotemporal blocks can change significantly since the temporal filters still need to be learned (related to the problem of covariate shift [13] in training deep networks). We can mitigate this effect by initializing all the temporal filters to the identity matrix, which will reduce the

video model to the image model. However, it is still possible that any slight change to the temporal weights may result in significant distribution changes to the spatiotemporal block, which can result in complicated optimization.

Similar to the motivation of ResNet [12], and different from R(2+1)D [26], we propose to alleviate this issue by adding a residual connection (as shown in Figure 5b) and initializing the temporal filters to zero. This effectively allows our model to learn the temporal filters gradually, and to have the "fallback" option to use just the spatial filter outputs when temporal information is not useful. We empirically found that adding the residual connection is crucial to have good performance, as detailed in Section 4.

The video model includes the same domain adaptation and attention components as earlier. The loss for the video domain in Figure 4 is

$$L_{video} = \mathbb{E}_{x^V}\big[-\alpha^V \cdot y^V \cdot \log(C(E(x^V)))\big] \quad (4)$$

which has the effect of ignoring or downweighting irrelevant video chunks in a soft way. The combined loss is similar to the image loss

$$\min_{\theta_E,\theta_C}\max_{\theta_D} L_{video}(E,C) + \beta L^V_{domain}(E,D) \quad (5)$$

where β is a tradeoff parameter.

3.7. Training

Putting all the pieces together, we first learn an image model (shown in Figure 3) using web images, web video frames, and target video frames as inputs. Each input is fed into a 2D CNN where we extract embeddings that are used to compute the domain adaptation and weighted classification losses. We then learn a video model (shown in Figure 4) by initializing the spatial filters from the learned 2D CNN and continue learning temporal filters from the videos. Similar to the image model, we use a 3D CNN to extract embeddings which are used to compute the domain adaptation, and weighted classification losses.

4. Experiments

4.1. Data

We evaluate our model on a standard benchmark for video classification, UCF-101 [24], and a larger dataset, Kinetics-400 [14]. UCF-101 consists of about 13K video clips for 101 action categories while Kinetics-400 is a larger dataset of 300K video clips for 400 action categories.

Similar to previous webly-supervised approaches [7, 9], we used standard image search engines (Bing and Google) to collect between 800-900 images (using the "photo" filter in the query) and YouTube to collect between 25-50 videos for each category. For our UCF experiments, the whole dataset consists of about 200K images and video keyframes. We follow the same process for Kinetics and collected about 400K images and video keyframes.[1]

Since UCF and Kinetics videos are both drawn from YouTube, it is possible there may be overlap with the webly-supervised images and videos we collected. To remove any potential overlap, we compared CNN embeddings extracted from video keyframes in the UCF/Kinetics videos and compared them to embeddings from the web images and videos. We then removed any web image or web video containing an embedding that had cosine similarity above a threshold (we used 0.9) with any UCF/Kinetics keyframe embedding. This process removed about 5% of the collected videos.

4.2. Implementation

We used ResNet-34 [12] as the base network for all experiments. Every image is resized such that the shorter dimension is 256 and then a random crop of 224x224 is extracted. For videos we first resize the video in a similar manner and then use the Hecate [23] tool to extract keyframes and video chunks. For each video chunk, we extract 24 frames, sampling every other frame to obtain a volume size of 12x224x224x3 per chunk. We use a batch size of 32 for images and 10 for videos.During training we take a random crop with the given volume size; during evaluation we take a center crop. We use a value of $\beta = 0.5$ and set T to be $0.9 \times$ batch size. All models are coded in PyTorch [19] and trained using stochastic gradient descent with momentum. We use a held out validation set (20K for UCF-101 and 40K for Kinetics) to choose model hyperparameters. The 2D CNN encoders and classifiers in the image model (Figure 3) and the 3D CNN encoders and classifiers in the video model (Figure 4) have tied weights.

4.3. Results

Our initial hypothesis was there may be a domain difference between images and videos that may be reducing the effectiveness of the model. To test this hypothesis, we trained a binary classifier (using ResNet-34 [12]) to distinguish between web images and web video frames and found that the classifier was over 99% accurate. We hypothesize that compared to web images, web videos tend to be lower resolution and may contain motion blur and compression artifacts not typically found in web images.

Next, we show an example of the weights learned by attention using a batch size of 32 in Figure 6. The weight (α in Equation 2) is shown for each web image along with the category of the image (the weights sum to 1). Images that are cartoon-like or contain excessive text tend to receive lower weight since they appear less similar to images from the target domain (UCF-101 [24]). Failure cases can be observed for the "CliffDiving" and "Drumming" images

[1] Note that we did not notice a significant improvement when using more data and we wanted to reduce computational overhead.

Input	Arch	Features	Accuracy (%)
I	S	App	62.9
F	S	App	57.9
I + F	S	App	70.1
I + F (A)	T	App	71.4
I + F (DA)	T	App	72.2
I + F (A + DA)	T	App	72.6
V	S	App + Temp	72.6
V (A)	D	App + Temp	74.0
V (DA)	D	App + Temp	74.3
V (A + DA)	D	App + Temp	**74.9**

Table 1: **Ablation study on UCF-101**. We evaluate the image and video models as well as the DA and attention components. We show the top-1% performance of each model averaged over 3 splits of UCF-101 [24]. Abbreviations are I: web image, F: web video frame, A: attention component, DA: domain adaptation component, V: web video, S: single branch (standard 2D CNN), T: triplet branch, D: dual branch (Siamese), App: appearance, Temp: temporal.

Input	Arch	Features	Accuracy (%)
I + F	S	App	39.6
I + F (A)	T	App	41.8
I + F (DA)	T	App	41.9
I + F (A + DA)	T	App	42.3
V	S	App + Temp	42.2
V (A)	D	App + Temp	42.5
V (DA)	D	App + Temp	42.5
V (A + DA)	D	App + Temp	**42.8**

Table 2: **Ablation study on Kinetics.**. We evaluate different model components and show the top-1% accuracy of each model. The abbreviations are the same as in Table 1.

in the last row. These images look reasonable but may have received lower batch score due to the extreme perspective and atypical color palette, respectively. Also attention does not help for images with the wrong semantic category (e.g. "CliffDiving" in the first row).

Table 1 shows ablation study results on UCF-101 [24]. For each row, the accuracy is averaged over the 3 splits of UCF-101. The inputs correspond to **I**: web images, **F**: web video frames, **I + F**: web images and video frames together, **V**: web video chunks. In addition, we train on the different model components **A**: the attention component, **DA**: the domain adaptation (adversarial) component, **A + DA**: both components. The model architectures correspond to **S**: single branch (i.e. a standard 2D CNN), **D**: dual branch (i.e. a Siamese network) corresponding to the image model, **T**: triplet branch corresponding to the video model. Lastly the features correspond to **App**: appearance (image) features, **Temp**: temporal (video) features, **App + Temp**: both appearance and temporal features.

We can see that a model trained with images and video frames together (I+F) outperforms a model trained with image (I) and video frames (F) separately. We verified that simply adding more images or video frames did not improve performance. Next, we can see that adding the domain adaptation (DA) and attention (A) components separately helps improve performance by a small amount, but both components together leads to the best image model, I+F(A+DA), at 72.6% top-1 accuracy.

The next step is to initialize the video model using the spatial weights of the image model, and then continue training on web videos. The video model V has an accuracy of 72.6%, which is the same as the image model accuracy. This may be due to irrelevant frames and noise present in the web videos that are unaccounted for. Similar to the image model, adding the attention and domain-adaptation components separately leads to a small improvement but adding both components together leads to the best performance of 74.9% top-1 accuracy on UCF-101 for the video model V(A+DA).

We also explored a couple variations of training the video model V. We first initialized V from ImageNet [5] spatial weights rather than the image model, which resulted in an accuracy of 59.1%. This drop in performance compared to model V (from 72.6% to 59.1%) may vindicate our two-step approach of training an image model based on web images first, since training on web videos directly led to worse performance. In addition, we explored a variation of the video model V which *does not* use the residual connection (corresponding to Figure 5a). This model achieves an accuracy of 70.3% which is significantly lower than the accuracy of V at 72.6%, which may indicate that adding the residual connection is crucial in our approach.

We compare our approach to previous work on UCF-101 in Table 3. Among webly-supervised approaches, we are competitive with the state-of-the-art LeadExceed [9] model at 76.3% vs 74.9% for our model. LeadExceed requires 5 stages of model training/refinement steps, while our model unifies classification and filtering, and requires only 2 stages. Thus our model simplifies the training procedure at the cost of a small drop in accuracy (about 1.4%). We note there is still a large gap between webly-supervised methods and the state-of-the-art methods which directly use the UCF training data (which is curated) and other features such as optical flow.

We also evaluate on the larger Kinetics [14] dataset. The results in Table 2 show similar improvements as the image model, by adding the attention and domain adaptation components, leading to the best performance of 42.8% accuracy. We also compare against leading methods in Table 4 and note that there is a large gap between our webly-

Figure 6: **Attention Weighting.** For a web image batch, we show the weights (α in Equation 2) for each image and the category of the image (the weights sum to 1). Images with lower weight in the last row tend to be more cartoon-like or contain excessive text while images with higher weight tend to be more representative of the action. Images such as "CliffDiving" and "Drumming" in the last row appear reasonable and can be considered failure cases since they are assigned lower weight.

Approach	Type	Pre	Train	Acc(%)
UnAtt [16]	App	IN	Web	66.4
Webly [7]	App	IN	Web	69.3
LeadExceed [9]	App + Temp	IN	Web	76.3
Our model	App + Temp	IN	Web	74.9
C3D [25]	App + Temp	K	UCF	82.3
2Stream [21]	App + Temp	IN	UCF	88.0
R2D-2S [26]	App + Temp	K	UCF	97.3
I3D-2S [3]	App + Temp	IN+K	UCF	**98.0**

Table 3: **UCF-101 Results.** Comparison to several top approaches on UCF-101 [24]. Abbreviations are App: appearance, Temp: temporal, Pre: pretraining data, Train: training data, Acc: top-1 accuracy, IN: ImageNet, K: Kinetics.

Approach	Pretrained	Training	Acc(%)
C3D[25]	ImageNet	Kinetics	57.0
2S [21]	ImageNet	Kinetics	61.0
R2D-RGB [26] 2S	Sports-1M	Kinetics	75.4
I3D-2S[3]	ImageNet	Kinetics	75.7
NL I3D [31]	ImageNet	Kinetics	77.7
SlowFast [6]	None	Kinetics	**79.8**
Our model	ImageNet	Web	42.8

Table 4: **Kinetics-400 Results.** Comparison to popular approaches on Kinetics [14] for top-1% accuracy on the validation set. The Two-Stream model is abbreviated as 2S.

supervised approach and state-of-the-art. State-of-the-art approaches such as SlowFast [6] use multiple spatial crops to improve video-level accuracy, which we can incorporate in the future. We are not aware of other webly-supervised approaches evaluated on Kinetics.

5. Conclusion

We presented a new webly-supervised approach for video classification using noisy web images and videos. Our model proceeds in two stages by first learning spatial filters of a 2D CNN from images, transferring the learned spatial weights to the 3D CNN video model, and continuing learning temporal filters with videos. To address the domain gap between the web and the curated data, as well as between images and video frames, our model incorporates an adversarial component to learn a domain-invariant feature representation between source and target domains. We also account for noise in web data using a novel attention component. We demonstrated performance competitive with state-of-the-art for webly-supervised approaches on UCF-101 [24] while simplifying training, and also evaluated on the larger Kinetics-400 [14] for comparison.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 4

[2] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 3

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CVPR*, 2017. 3, 8

[4] J. Chen, Y. Cui, G. Ye, D. Liu, and S.-F. Chang. Event-driven semantic concept discovery by exploiting weakly tagged internet images. In *ICMR*, 2014. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 7

[6] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *CoRR*, 2018. 3, 8

[7] C. Gan, C. Sun, L. Duan, and B. Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *ECCV*, 2016. 2, 5, 6, 8

[8] C. Gan, C. Sun, and R. Nevatia. Deck: Discovering event composition knowledge from web images for zero-shot event detection and recounting in videos. In *AAAI*, 2017. 2

[9] C. Gan, T. Yao, K. Yang, Y. Yang, and T. Mei. You lead, we exceed: Labor-free video concept learning by jointly exploiting web videos and images. In *CVPR*, 2016. 2, 6, 7, 8

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(59):1–35, 2016. 1, 3, 5

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 2, 3, 4

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4, 6

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[14] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. 6, 7, 8

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 1

[16] J. Li, Y. Wong, Q. Zhao, and M. S. Kankanhalli. Attention transfer from web images for video recognition. *ACM Multimedia*, 2017. 2, 8

[17] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei. Label efficient learning of transferable representations across domains and tasks. In *NIPS*, 2017. 3, 4

[18] S. Ma, S. A. Bargal, J. Zhang, L. Sigal, and S. Sclaroff. Do less and achieve more: Training cnns for action recognition utilizing action images from the web. *Pattern Recognition*, 2017. 2

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[20] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 3

[21] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 8

[22] B. Singh, X. Han, Z. Wu, V. I. Morariu, and L. S. Davis. Selecting relevant web trained concepts for automated event retrieval. In *ICCV*, 2015. 2

[23] Y. Song, M. Redi, J. Vallmitjana, and A. Jaimes. To click or not to click: Automatic selection of beautiful thumbnails from videos. In *CIKM*, 2016. 6

[24] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. 2, 6, 7, 8

[25] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 3, 8

[26] 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. 1, 3, 5, 6, 8

[27] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015. 3

[28] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 1, 3

[29] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008. 2

[30] L. Wang, Y. Xiong, D. Lin, and L. Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *CVPR*, 2017. 2, 3

[31] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 8

[32] J. Zhang, Z. Ding, W. Li, and P. Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *CVPR*, 2018. 3, 4

[33] B. Zhuang, L. Liu, Y. Li, C. Shen, and I. D. Reid. Attend in groups: a weakly-supervised deep learning framework for learning from web data. In *CVPR*, 2017. 3, 5