CAST: Character labeling in Animation using Self-supervision by Tracking

Oron Nir[1,2] , Gal Rapoport[1] , and Ariel Shamir[1]
[1]Reichman University , Israel, [2]Microsoft Corporation

Abstract

Cartoons and animation domain videos have very different characteristics compared to real-life images and videos. In addition, this domain carries a large variability in styles. Current computer vision and deep-learning solutions often fail on animated content because they were trained on natural images. In this paper we present a method to refine a semantic representation suitable for specific animated content. We first train a neural network on a large-scale set of animation videos and use the mapping to deep features as an embedding space. Next, we use self-supervision to refine the representation for any specific animation style by gathering many examples of animated characters in this style, using a multi-object tracking. These examples are used to define triplets for contrastive loss training. The refined semantic space allows better clustering of animated characters even when they have diverse manifestations. Using this space we can build dictionaries of characters in an animation videos, and define specialized classifiers for specific stylistic content (e.g., characters in a specific animation series) with very little user effort. These classifiers are the basis for automatically labeling characters in animation videos. We present results on a collection of characters in a variety of animation styles.
Code and resources are available at: `https://github.com/oronnir/CAST`.

CCS Concepts

• Imaging and Video → Video Summarization; • Methods and Applications → Artificial Intelligence; Computer Vision; Neural Nets;

1. Introduction

Recognizing and labeling characters in a given video is an important task for many applications. It can empower media companies to search, analyze, and reuse content in videos. Many current solutions for real-life videos are based on face detection or people detection. However, in the animation and cartoon domain this task is more challenging. First, there is a wide diversity of styles and genres in animation such as Manga, hand-drawn and CGI movies that can have significant differences in appearance. In addition, they are all very different from real-life videos. Trying to build one semantic representation for all animation styles or relying on representations constructed for real-life videos is problematic. Second, characters in animation videos could be anything from a person to an animal or even an object like a talking candelabra. Relying on face or people detectors is insufficient. Third, the same character can change color, texture, style, and shape, even in the very same scene (see Figure 1). Such variability means that the semantic representation of a character cannot be based on appearance alone, cannot use natural images learned representations and must include several possible instance variations of the same character.

In this paper we suggest learning a semantic representation suitable for character identification and labeling specific to a given animation style using self supervision. As a first step towards gen-

Figure 1: *The same character in animation can change color, appearance and even shape. Our challenge is to build a representation that can maintain its semantics and map all visual depictions to the same identity.*

eralization, we label an animated video dataset in various styles and leverage it to refine an object detector and to define a base latent representation for animation by refining a classifier in a supervised setup. This step is performed once to better capture the characteristics of animated media as opposed to natural videos for detection. On the other hand, character labeling demands precision for identification, and therefore a more style-specific representation. To adapt the representation towards specific animation styles and move different instances of the same character closer in the semantic space we introduce a self-supervised learning method. The method takes an input video (e.g., one episode of a series or a full feature film), segments its shots, and runs the base object detector on this video. A multi-object tracker (MOT) is later applied per shot to follow all characters in the shot through changes in appearance and shape. Next, triplets of examples are sampled based on the characters' tracklets, and are leveraged to refine the learned representation using contrastive loss. The underlying hypothesis is that two detected characters within the same tracklet share identity while two that appears in the same frame do not. This alters the semantic space towards representing the specific animated style, where distances convey the identity of characters and not simply appearance differences.

To illustrate the effectiveness of the new representation space we use clustering to build more coherent and meaningful clusters of examples for each character. We demonstrate the application of this space for long-tail dictionary building of characters in a video, where many characters, even if they appear sparsely, populate the dictionary. We also use this representation to define more effective animated character classifiers, and demonstrate their use for dense identification and labeling of characters on unseen test-videos. Hence, we call our method CAST: Character labeling in Animation using Self-supervision by Tracking.

2. Related Work

Recent advancements in deep neural networks applications for various computer vision tasks like object detection and facial recognition have matured to solve problems at an industrial scale. However, specific domains with less available data, like animation and art, many times require a different approach [OON*18, YNS19, TOYA18].

Animated character recognition has been addressed with machine learning models before. Yu et al. [YS11] have suggested a fuzzy diffusion distance as a cartoon similarity metric for recognition and clustering purposes achieving up-to 77% accuracy, using Kuhn-Munkres algorithm, on a proprietary dataset with 60 classes. Zhang et al. [ZHEL*13] aimed at detecting IP violations and piracy with Scalable-Shape Context and Hough Voting. Nguyen et al. [NRB17] suggested a convolutional neural network-based (CNN) comic characters detector for a comic book recognition problem. They focus on a binary task of character detection and have reported 92% precision. They have based their detector on the YOLO-V2 detector by Redmon & Farhadi [RF17]. Our work identifies characters in video, that introduces higher character variance compared to static comic books. Chu et al. [CL17] have suggested a CNN based model for detecting Magna faces in grey scale. They have addressed this challenge and taken the

approach of using a joint CNN for bounding box regression and non/character classification on candidate regions extracted using selective search. Their method reached 66% F1 score on the MAGNA109 dataset [FOY*16]. Ogawa et al. [OON*18] have used an SSD300 architecture for the same detection problem on this dataset and reached an average precision of 67% and 79% on face and body, respectively.

Landmark detection in art was also addressed by [YNS19, JAA18] and emotion recognition in animation by [Sta18, YZL19]. Caricature recognition was benchmarked by Huo et al. [HLS*17] suggesting that the artistic aspects embedded in a caricature challenges the approach of augmenting real-life images as they impose a significant domain difference.

Human character identification that works on realistic videos usually rely on detection and clustering of faces or humans [ZXLH09, ACT16, AWS*18, CZD*19]. Naming of characters is usually done by searching the web or using social media. Since animation resources and datasets are less common, the only manual part in our method is naming characters for identification.

Clustering of real-life faces is yet another well researched field. Zhu et al. [ZWS11] have suggested a rank order distance to group similar instances of the same person. Schroff et al. [SKP15] have introduced FaceNet and the triplet-loss for projecting images onto a latent space that quantifies similarity in a supervised-learning manner. Recently, Somandepalli et al. [SHN21] used tracking of faces in a photo-realistic video, followed by clustering and verification using MvCorr [SKJ*19] and Improved Triplet [ZGW16] to adapt available face representation data to perform better on racially diverse images following [STSS19]. Aneja et al. [ACF*16] have suggested DeepExpr model for facial expression recognition for multiple styles. Tsubota et al. [TOYA18] have used the MANGA109 dataset for manga face clustering. The approach they took for generalization of deep embeddings adaptation on the inferred manga comics was based on deep metric learning (DML) inspired by the work of Zhang et al. [ZLLT16]. They reached a normalized mutual information (NMI) of 71% and accuracy of 64% based on manga domain specific priors e.g., co-occurrence per comic scene. The 'k' number of characters was assumed to be given and was not estimated. The underlying CNN architecture was ResNet50 [HZRS16]. An ablation study suggested that without the page information the NMI drops to 67% and the accuracy to 58%. This indicates the importance of a temporal analysis in our challenge. Shen et al. [SEA19] have addressed a similar problem, discovering visual patterns in art collections with spatially consistent feature learning also known as ArtMiner. They have suggested a similarity metric based on ResNet18 for artwork elements to automatically recognize copied pieces of work created in different styles e.g., oil, pastel, water-color, etc. They have reported 88.5% accuracy on LTLL [FTT15] and 85.7% mAP on Oxford5k [Phi07]. Recently, [ZZR*20] have published iCartoonFace dataset for face detection and identification in animated images. While anthropomorphized faces tend to have the same facial features people have, an animated character's body shares less common attributes and perhaps makes the generalization a more challenging task.

The closest prior work to ours is [SKGN18] who built a system that aims at finding the main cast of a given movie which
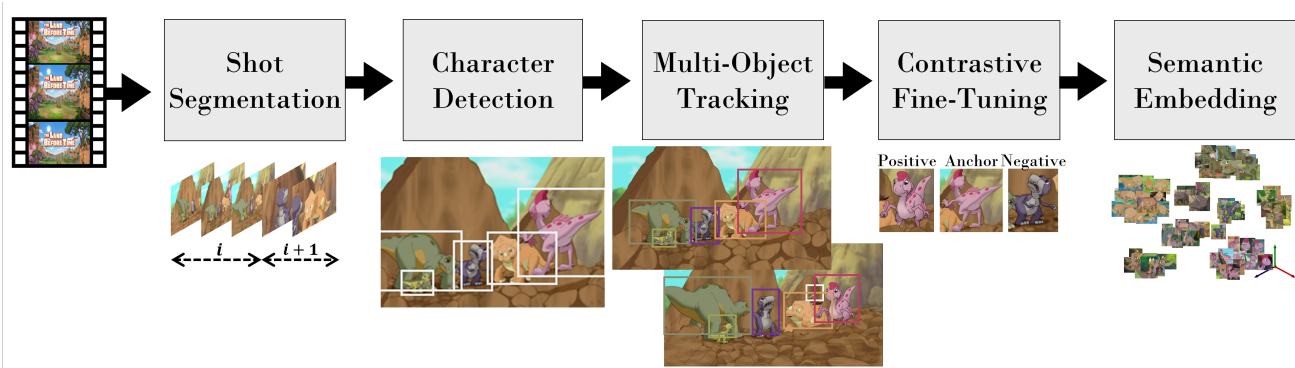
Figure 2: *CAST self-supervised representation learning pipeline includes: automatic shot segmentation, detection, multi-object tracking, and contrastive-based refinement to define the unique semantic embedding for an animation style.*

is termed 'unsupervised discovery of character dictionaries' . Our work not only builds the dictionary but allows to create classifiers that can recognize and index characters at frame-level and generalize to additional episodes of a given series. Their work applied the MultiBox detector on uniformly sampled frames followed by an object tracker, ImageNet FC7 layer for embeddings and Afinity Propagation for clustering. Their overall dictionary F1-Score is 72% and 70% cluster purity on a dataset of eight videos of SAIL AMCD. To evaluate the over/under clustering performance they've measured the median number of clusters per character which was 3. We consider this paper as our baseline for dictionary building and compare our results. Kim et al. [KLS*20] have addressed the same task. Their main contribution was a character detector that adapts the animation style with a region-based Faster R-CNN following the work of [WCGV19]. The rest of their algorithm aligns with [SKGN18]. However, since their ABCD data set is proprietary and they do not report cluster purity and median clusters per character it was difficult to compare to.

3. Representation Learning

An overview of the CAST representation pipeline is represented in Figure 2. First, the video is segmented to shots according to the method of Hua et al. [HLZ04]. Next, frames are sampled consistently on which our character detector is applied, providing character bounding box proposals. These proposals are then tracked while those who share a tracklet should appear closer in the embedding space than other proposals in the same shot for they belong to different characters. A set of triplets examples are then sampled and used to further refine the basic mapping network for the specific style of animation. This representation is later used for clustering, dictionary discovery and classifier definition of characters in this animation style.

3.1. Data Acquisition

Building a CNN-based semantic representation must address both the scale and diversity of the data in the domain. Thus, we gathered data sets of cartoon and animated content from various sources such as media companies and the web. For the basic TRAINING dataset,

we collected 174 videos in 61 different styles including Anime, CGI, 2D cartoons, and more (see supplemental material). Examples of series collections in this dataset are: Blender (3D CGI), legacy Looney-toons videos published under public domain (2D hand drawn in color as well as gray-scale), Manga of various styles, Indian/Korean animated films, and more. From these videos more than 118K frames were extracted and annotated manually. Characters in each frame were marked with a bounding-box and named (using Microsoft's UHRS - like [FCH*19]). This resulted in more than 250K ground-truth bounding-box instances of 549 distinct characters.

We used this basic TRAINING dataset to define two neural networks: the first was trained to detect basic proposals for animated characters in animation videos while the second was trained as the basic mapping of proposals into a semantic embedding space, which is later refined using self-supervision for each specific animation style representation.

For testing the domain-adapted detector and selecting a clustering algorithm, we used a second TEST data set. This set contained 50 videos with a total length of 24 hours and 681 characters. This dataset was collected from YouTube with various styles e.g. Gaming, amateur and professional productions content, and various other genres.

For evaluating the dense character identification application, presented later, we collected a third, EVALUATION, dataset from seven different cartoon series, two episodes each. One episode was used to guide the self-supervision and train classifiers, and the other was manually labeled and used for testing the classification results based on the first episode. This data set is 11 hours long and includes 49 characters. More details on the three sets can be found in Table 1. Note that the three datasets do not contain any overlap i.e., no character, episode or series appears in more than one dataset. We plan to publish a labeled dataset for future work comparison.

We also used the SAIL AMCD dataset [SKGN18] to evaluate CAST for unsupervised discovery of character dictionaries and compare our results to [SKGN18]. This set contains eight evaluation videos that are full-length feature films of different animation

DATASET ROLE	STYLES[#]	VIDEOS[#]	KEYFRAMES[#]	HOURS	CHARACTERS[#]	BOXES[#]
TRAINING	61	174	118,751	65.6	549	257,706
TEST	50	50	38,821	24.1	681	90,325
EVALUATION	7	14	17,583	11.0	49	50,300

Table 1: *Statistics for the* TRAINING, TEST, *and* EVALUATION *CAST datasets.*

styles. The videos were trans-coded to standard HD and sampled at 4 FPS.

3.2. Basic Animated Character Detection

Due to the high variance in characters' appearance, and the fact that animated characters are not necessarily humans, a simple person detector for character proposals will not suffice. Our base detector is the YOLO V2 generic object-detector architecture [RF17] which was originally trained on ImageNet [DDS*09] with millions of images. We use our TRAINING set to fine-tune YOLO's Person class for 20 epochs and use it as a binary classifier, character vs. non-character. The person detector was chosen as many animation characters still tend to have a human-like attributes. Data augmentation methods were applied to avoid overfitting randomizing both vertical and horizontal flip as well as resize with factor range [0.5, 2.0]. We also tried to fine-tune a different architecture based on Faster-RCNN [RHGS15], which is a two-pass CNN. However, we found our architecture, based on YOLO's fast low compute one-pass detector, to be faster and produced higher average precision. We compare this detector to the following alternatives: using the YOLO V2 single class person detector, and using all combinations of the classes: Person, Animal, Mammal, and Toy as a detector. The detectors' effectiveness was evaluated in terms of Average Precision (AP) at a 50% Intersection over Union (IoU) using the TEST dataset. Our basic character detector has reached an AP of 61% significantly outperforming the YOLO V2 person detector (AP=50%) and the combined classes version (AP=50%) in a wide range of animation styles.

3.3. Base Embedding Network

Feature maps in deep layers of networks such as VGG [SZ15] and ResNet18 [HZRS16] have been shown to represent high-level semantic meaning in many previous works. However, such models were trained mostly on natural images, therefore could misinterpret animated content. Similar to the above approaches, to learn a representation suited for our domain, we use our TRAINING dataset to fine-tune classification networks and use the feature map of its deepest hidden layer as the representation of the base embedding space. The TRAINING set containing 549 different characters was heavily long tailed: 40% of the characters have less than 50 samples. Therefore, we have balanced the dataset during training by repeating rare characters' images, such that per epoch, each character has at least 50 samples. To avoid overfitting, data augmentation transformations have been applied on the training data at random using the following methods: crop resize, Affine transform, color jitter, and horizontal flip.

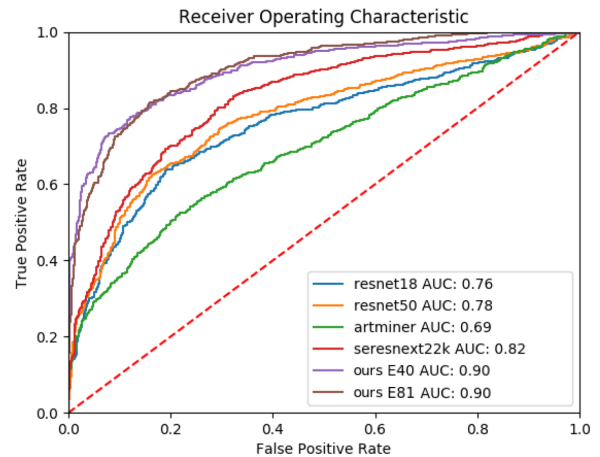We compared different configurations of three different ar-

Figure 3: *RoC curve per basic semantic embedding backbone architecture: ResNet18, ResNet50, ArtMiner, SEResNeXt22k (the original network), our network fine-tuned for 40 epochs as well as a further trained version to illustrate the convergence.*

chitectures: ResNet18 [HZRS16], SEResNeXt [HSS18] and Art-Miner [SEA19]. SEResNeXt was found to perform best and therefore the last average pooling layer was used (20th squeeze and excitation block). Yielding vectors of size 2,048, as the basic semantic representation for animation characters bounding boxes.

To evaluate this representation, we measured how well this representation conveys semantic similarity. We built a test-set by creating a balanced dataset of 600k pairs of similar and dis-similar images of characters from the TEST dataset. We defined a similarity measure for such pairs to be the cosine similarity of our embeddings of the extracted boxes. The pairs were ordered according to their similarity and addressed in a binary classification design with a RoC curve. In Figure 3 we plot the curves of the various tested alternative representations. Our representation, using an SEResNeXt back-bone, has outperformed all alternatives on the test data. We further found that training the network for more than 40 epochs does not provide much gain.

3.4. Self-Supervision

Similar to previous works, e.g. [KLS*20], we strive to refine the semantic space for each specific animation style. Given a specific animation style (for instance, an episode of an animation series), we fine-tune the base embedding network towards this style, using a novel self-supervised approach which combines multi-object tracking with contrastive learning (triplet margin loss [SKP15,BRPM16]
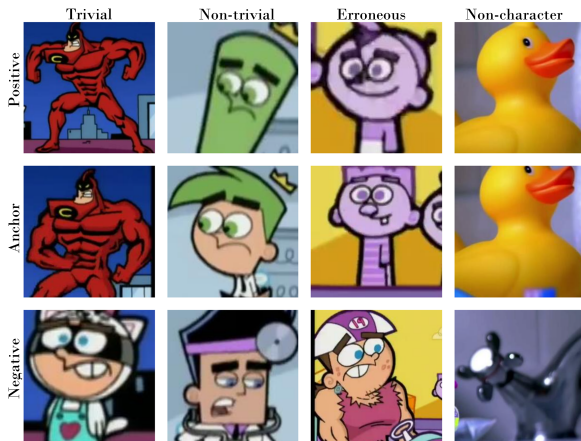
Figure 4: *Triplets examples illustrates the Positive, Anchor, and Negative triplets. The method aims for trivial and non-trivial examples while it is still resilient for erroneous and non-character examples.*

Figure 5: *Visualizing the first two principal components projection of our semantic embedding space before and after the Self-Supervision. We demonstrate the effectiveness of our method as a representation learning technique.*

with margin=1.0). Our main hypothesis is that two samples from a single tracklet can be used as positive and anchor examples (coming from the same character but in different frames), while another character proposal from the anchor's frame can be used as a negative example (coming from the same frame but different character).

We sample 10k triplets and refine the base embedding network for 10 epochs using AdamW [LH19] ($LR = 2 \cdot 10^{-5}$, batch size=20, $\lambda = 10^{-4}$, $\gamma = 0.1$, $\hat{m}_t = 0.9$). This setup applies for any specific animation style in PyTorch [PGC*17]. Tracking is performed with a classic multi-object tracking (MOT) approach of max-flow-min-weight algorithm inspired by the works of Zheng et al. [ZLN08] and Wang et al. [WWW*19]. A sparse network based on detection is constructed, Next, an optimization algorithm finds the optimal flow in the graph that maximizes its MAP representation. Lastly, the tracks are derived from the flow function using a greedy shortest weighted path algorithm as described next.

To construct the network we include an artificial source and sink nodes, and each detected bounding box is considered as two nodes ('in' and 'out') with a linking edge weighted by its detection confidence. Skip connections are added every FPS frames to overcome temporary occlusions. The edge weight between detections of neighbor (or skip) frames is weighted by the following six factors: (1) Time gap proximity w.r.t. the sample FPS, (2) $P_{IoU} = 0.5(IoU + 1)$ so disjoint boxes could potentially be linked, (3) Scale difference ratio, (4) Euclidean distance in pixels of their centers, (5) semantic similarity using the original embeddings, and (6) scale-weighted center distance as characters which are closer to the camera may have higher relative angular velocity. The six factors above are aggregated together using a geometric weighted average into a match likelihood measure. Their corresponding weights are (1.0, 1.5, 1.5, 2.0, 3.5, 4.5). Each edge in this network is considered as a unit capacity network and solved with the interior-point algorithm as a Linear Programming (LP) representation [AA00]. Since the constraints matrix is totally unimodular the LP solution is also
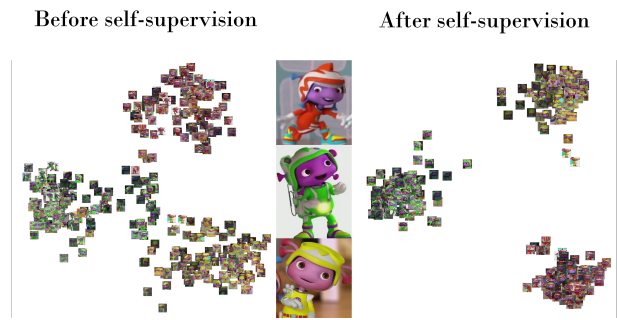
a valid integer programming solution [ZLN08]. Each tracklet solution per shot is greedily derived from the flow function using a weighted DAG shortest path algorithm and its MAP tracklet probability is considered to be its significance. Tracks significance filter threshold per shot is defined to be 10% of the shot's most significant tracklet.

The triplets gathering is performed by sampling a shot with at least two tracklets, out of which a frame is picked, with at least two proposals (bounding boxes). The first is used as an anchor while the second is used as the negative example. Then, from the anchor's tracklet, a third proposal is randomly picked as the positive example. Figure 4 illustrates the some triplets types. Many standard triplets are gathered using this method where a character preserves its appearance in the positive pair and contrasted with a different character. However, this method also captures non-trivial examples in which the character itself changes its appearance or shape. The two right columns in Figure 4 illustrate erroneous triplets that can also be gathered using our method: the first is the results of errors in tracking causing a wrong positive pair, and the second is an error in the detector where non-characters (objects) are considered characters. However, due to the large amount of triplets gathered our mapping is resilience to such errors.

4. Semantic Clustering

A strong indicator for a meaningful representation is the ability to cluster items in the semantic embedding space. Instances of the same animated character, even visually different, should all map to a relatively close position in the embedding space, while instances of different animated characters, even if they are visually similar, should be placed further away in the embedding space. Therefore, if we cluster the bounding boxes in the embedding space, the results should be coherent clusters of characters.

First, we demonstrate the effectiveness of our base embedding network. We conducted an experiment, where we split the TEST dataset to six subsets of various characters and used k-means clustering in the embedding space. We use cluster purity [MRS08] as our measure as opposed to clustering evaluation measures like NMI

Self-Supervision	Clusters [#]↓	Pure [%]↑	Characters per video[#]↑	Labeled boxes per video[#]↑
Before	28.7	70	7.0	396.2
After	**20.0**	**90**	**13.1**	**599.3**

Table 2: *Clustering ablation study on our* EVALUATION *dataset. The better algorithm has less **Clusters** to label, higher **Pure** percentage, more **Characters per video**, and as many **Labeled boxes per video**.*

and F-score, and evaluate the performance of the clustering quality as a function of the number of clusters. We compared our embedding with SEResNeXt22, ResNet18, ResNet50, and ArtMiner for different values of 'k'. This comparison highlighted the superiority of our base embedding over the alternatives as for every value of 'k', the purity of clusters in our base embedding was significantly higher. The full details of this experiment can be found in the supplemental materials.

Next, we compared several clustering algorithms, namely k-means, mean-shift, modularity maximization, affinity propagation, DBSCAN agglomerative clustering, and spectral clustering on our TEST dataset. We use purity [MRS08] and K-metric, which is the geometric mean of the class purity and the cluster purity over the contingency matrix [PRNZ*11]. The two best performing were DBSCAN and agglomerative clustering. They yielded a median cluster purity of 96% and K-metric of 40%. We selected DBSCAN as the basic clustering algorithm as it handles noisy character proposals filtering inherently. The hyper parameters that govern the confidence score for filtering were determined by grid search using the TEST dataset to be $\lambda_1 = 0.275, \lambda_2 = 0.4$. To optimize clusters purity, we binary search for the ε parameter of DBSCAN so that it maximizes the Silhouette times the non-noisy proposals in the range $25 \leq k \leq 60$. This unsupervised criterion balances purity and cluster size.

Since we optimized for *recall* during the proposal detection stage, there are still many outliers that are non-characters among the proposals. We remove outliers inside each cluster and remove low-confidence clusters altogether. This filtering is based on a confidence score that was calculated as a product of the detection's confidence and a squared exponential kernel of the sample's distance to the cluster center:

$$k_{\text{SE}}(x,\mu) = \exp\left(-\frac{(x-\mu)^2}{\lambda_1^2}\right)$$

where μ is the cluster's median, x is the sample, and λ_1 is a hyper parameter. A sample is considered an outlier when its confidence score is less than the cluster's $Q_{25} - \lambda_2 \cdot IQR$ where Q_{25} is the 25th percentile, IQR is the interquartile range and λ_2 is another hyper parameter. Finally, clusters with average centers that are up-to 0.7 cosine-similarity are merged back together.

Our combined MOT and contrastive-loss based self-supervision allowed further representation enhancement. The performance analysis using the EVALUATION dataset shows that 90% of clusters created after the fine-tune stage were totally pure across all styles, while each character has a median of 1.2 clusters (lower

is better). This demonstrates a significant improvement over the base-embedding as the clustering algorithm has yielded 30% less clusters for the same video i.e., lower over segmentation. These clusters captures 13.1 characters per video instead of 7.0 i.e., 87% better while these clusters have higher purity. (See Table 2 and an illustrative example in Figure 5). As can be seen in Table 3, for all different style of videos in our EVALUATION set, the clustering quality measured using silhouette increases as a result of our self-supervision refinement.

5. Applications

5.1. Long-Tail Dictionary Construction

The first step towards automatic labeling of characters in animation movies is automatic characters cast discovery. In this task, the goal is to build a dictionary of characters appearing in the movie. For any given animation style, we first use our automatic method to learn the refined mapping tuned to this style. Then, for each new episode or movie in this style we use our detection pipeline to gather character proposals and map them to the semantic space. We then cluster and filter the proposals in this representation space. This representation promotes purity so that each cluster will contain proposals belonging to a single character and encourages only a small number of clusters (preferably one) representing the same character. Then, to build the dictionary we create an entry for each significant cluster.

Proposals Creation: To reduce the chance of a miss-detection, the character proposal detector is tuned to be *recall* oriented as False-Positive detection is recoverable downstream while False-Negative is not. Only bounding boxes with less than 20% confidence or having a size smaller than 2.5% of the frame are filtered out at this stage. This causes repetitions of semi-identical proposals, i.e., bounding boxes with highly similar content. Hence, we apply a screening procedure to remove duplicate proposal. Each proposal is represented by an Edge Directional Histogram (EDH) feature vector of size 116 [WZM12]. The 64-dimensional color and texture features ensures global similarity between two proposal and the $4 \cdot 13$ EDH features ensures spatial similarity with detailed constrains by edge. The EDH was computed using a Canny edge detector with a 7×7 Gaussian convolution kernel. Next, the cosine-similarity was computed between all pairwise proposals, and an undirected graph was built with proposals as nodes and cosine-similarity as edge weight. Edges of weight lower than 0.995 were pruned. Finally, cliques were found in this graph using [BK73, CK08], and each clique was aggregated into a single proposal.

Characters Selection: To build the dictionary we first cluster all candidate bounding boxes and filter them as described in Section 4. Next, we select every cluster and compute the median vector of all proposals in the cluster. We pick the proposal closest to the median as the representative and insert it to the dictionary. We report the performance of our algorithm for unsupervised discovery of character dictionaries both on our EVALUATION dataset and on SAIL-AMCD, comparing our results to [SKGN18]. Although [KLS*20] reported similar results to ours, their dataset was not available and they did not report the aggregated purity of their clusters.

Self-Supervision	#1	#2	#3	#4	#5	#6	#7	Avg.
Before	0.156	0.258	0.166	0.285	0.231	0.382	0.247	0.246
After	0.320	0.423	0.364	0.601	0.248	0.520	0.332	0.401
Silhouette gain↑	0.164	0.166	0.199	0.317	0.017	0.137	0.085	0.155

Table 3: *Self-Supervision introduces a significant gain of 0.155 (P-val=0.0047) to the Silhouette score on our* EVALUATION *dataset.*

SAIL AMCD Video /Method avg. score	Precision [%]↑	Recall [%]↑	F1 [%]↑	Purity [%]↑	Med. exemplars per character↓	Avg. exemplars per character↓	Additional characters [#]↑
Cars 2	76.9	75.0	75.9	98.1	1.00	1.286	13
Free Birds	100.0	90.0	94.7	98.6	1.00	1.360	15
Frozen	96.0	100.0	98.0	99.5	2.00	2.000	9
Dragon 2	95.0	91.7	93.3	94.7	1.00	1.444	6
Shrek Forever After	89.7	90.0	89.8	90.5	1.00	1.679	18
Tangled	86.4	77.8	81.8	84.2	1.00	1.714	5
The Lego Movie	85.2	100.0	92.0	95.0	1.00	1.522	11
Toy Story 3	93.5	77.8	84.9	99.8	1.00	1.483	11
CAST (ours)	**90.3**	**87.8**	**88.8**	**95.1**	**1.125**	**1.561**	**11.0**
Somandepalli et al.	81.0	65.2	72.2	70.3	3.00	N/A	N/A

Table 4: *Unsupervised Discovery of Character Dictionaries comparison on the* SAIL AMCD *test videos.*

On our CAST EVALUATION dataset, yields 16.8 clusters per video, when the average number of characters per video is 13. On average the cluster purity was 98.5%. The precision, recall and F1-score of these dictionaries are 91.8%, 83.1%, and 86.6%. A visual illustration of the power of self-supervision refinement can be seen in Figure 1. Not only that all the different manifestation of this character (including kid/toaster/cat/broccoli) were mapped to the same cluster, but this cluster, including 94 bounding boxes, was 100% pure.

Comparing CAST with the work of Somandepalli et al. [SKGN18] on the SAIL AMCD test set (see Table 4). CAST outperforms the state of the art in all metrics and indicates 16.6% improvement in F1 as well as 24.8% in purity. Moreover, CAST allows to build a much larger dictionary containing twice as many characters that were not chosen in the original evaluation as lead characters (see examples of dictionaries in the supplemental material).

5.2. Defining Classifiers for Characters

The second step towards automatic labeling of characters in animation movies is the construction of dedicated classifiers for each character in the movies. The key idea is to gather enough examples that are diverse enough for each character to build a training set for effective learning. Using CAST, the dictionary is presented to the user for naming. This allows us to use all the proposals in every named cluster as training data for classifiers. This also allows us to merge clusters when the same character is found more than once in the dictionary. Lastly, this also allows us to filter noisy clusters, where several characters appear in the cluster. We do this by presenting a number of examples from each cluster to the user and validating that they are all examples of the same character.

Using this method, a specialized training set for each character can be constructed with minimal user effort. These training sets are then used to train specialized classifiers as needed. Moreover, when processing a new video, the user can choose to apply the existing classifier model for automatic identification of some characters or add additional new characters by further naming clusters of unidentified characters.

Training Classifiers: All the named clusters are merged into one training-set that allows training a multi-class image classifier for the specific animation styles. In our experiments we fine-tune a state-of-the-art CNN classifier SEResNeXt [HSS18] for 40 epochs using typical augmentation operations such as rotations and mirroring.

To best train a mutli-class classifier, it is important to introduce an additional class of negative examples which comes from the same animation style. Typically, these examples are cropped from the background of processed images. Since our model is aimed at working in the wild on new series and new animation styles, we devised a method to create negative examples automatically per animation style. The base detector that marks characters bounding boxes on keyframes is biased for recall. Hence, our hypothesis is that all other pixels can be considered background. The challenge is to find large enough sub-frames that do not intersect with the characters' bounding boxes (examples available in the supplemental materials). Finding the largest empty rectangle (LER) is a known problem [AF86]. In practice, we devised a recursive algorithm that extracts more than a single empty rectangle per keyframe, not necessarily the largest ones, but with some minimal width, height, and area at a runtime of $O(n^2)$ where n is the number of proposals. For the algorithm and the proof see supplemental materials. The resulting classifier can then be used on other videos of the same style (e.g., other episodes of the same animation series) to classify known characters. Still, new characters can appear in such videos, as well as some known characters with different appearances. In
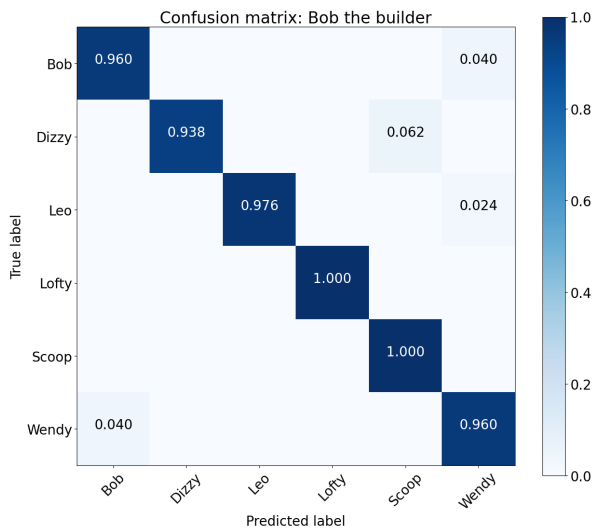
Figure 6: *'Bob the builder' confusion matrix (normalized rows).*

such cases, our method can be applied again with the same procedure of clustering and naming but only on the unknown proposals of any new video processed, to create new training examples and re-train the classifier further.

Experiments: To evaluate the effectiveness of CAST to define specialized classifiers for animated characters identification, we use the EVALUATION set that combines two episodes of seven different animation series. The animation styles include CGI, 2D, and Cutout techniques, with characters including humans, animals, dinosaurs, cars and tractors. For each series, we denote one training episode to create the dataset and train a classifier using CAST, and one test-episode, where several character proposals were manually selected and named for evaluation.

Statistics on CAST results for the seven train-episodes can be found in Table 5 and the supplemental material. As can be seen, CAST allows defining classifiers for animation characters by naming (or discarding) just a few clusters from the dictionary instead of manual labeling and filtering thousands of proposal bounding boxes. Cluster Purity is reported both before and after the noisy cluster filtering under 'General Purity' and 'Relevant Purity' respectively. 'Fireman Sam', staring four firemen figures which wear the same uniform sometimes clustered together and damage the general purity.

The results on the test-episode of each animation style can be found in Table 6. There are differences in the number of appearances of the different characters in the seven test-episodes. We randomly picked, on average, 22 instances of each character in the test-episodes, creating 1,742 characters of ground-truth test data. The total results yielded a mAP, F1-Score, and Accuracy of 92%, 88%, 89% respectively. The best performing image classifier was created for the series 'Bob the builder' with Accuracy and F1-score of 97.4% and 97.4% (see the confusion matrix in Figure 6, and the matrices of all series in the supplemental materials). The least performing image classifier was created for the series 'The Land Be-

fore Time', still achieving mAP and F1-score of 82.4% and 74.2% on 8 characters.

We further tested two series by annotating *all* proposal bounding boxes in the test-episodes of the series as ground truth. In this case, many proposals do not contain one of the characters and are marked as 'unknown'. In the supplemental materials we show the confusion matrices in these cases. The matrices show almost no confusion among characters, but there are some mis-classification between characters and non-characters. There are fewer false-positive examples where non-character proposals are classified as characters and more false-negative examples. Error analysis on the false-negative cases revealed that many of them contain character parts such as hands, legs etc. These proposals were identified and annotated as characters by the human annotator but are still a challenge for automatic classifiers. This may be an avenue of future research on identifying partial and occluded animated characters.

We confirmed an underlying assumption that training a classifier on different animations, consist of different characteristics, such as style, texture, colors, geometry as well as coarser and fine-grained characteristics, would yield different classifiers. By allowing a self-supervised training per animation, we demonstrated an overall improvement. Introducing a novel animation specific classifier framework, has significantly improved the classifier's quality metrics (Purity, F1-Score, etc.). Moreover, it resulted a substantial reduction of 56.6% in the number of clusters per character. That is, significant reduction in the annotations process.

Class-per-cluster vs. Class-per-character: The method we used above aggregates all clusters that share the same label into a unified class for training. An alternative design can create a class per cluster to train the multi-class classifier and consolidates the class name results after the prediction. For instance, if the character Bob from the series 'Bob the builder' has two dictionary entries, then their clusters will be used separately in the classifier training as Bob-1 and Bob-2, but both will be mapped to Bob after the prediction. The hypothesis behind this alternative design is that the feature space might not be connected in terms of the label distribution in the high dimensional space. Consolidating all labeled data into the same class may introduce an unnecessary error. Alternatively, the clusters themselves encode connected (or convex) similarity regions that can represent the character in some particular settings like wearing specific clothes. To test this hypothesis a two-tail paired t-Test was conducted to compare the F1-score change in the class-per-character baseline design vs. the class-per-cluster alternative hypothesis. The unit of analysis was a character using the EVALUATION dataset (N=49). We found that the our choice of merging clusters performed 5.5% better (higher) than the alternative (P-value=3.4%). To conclude, our learned embedding-model seemed to encode coherent regions in a way that a consolidated class design would leverage better than separated classes per cluster.

5.3. Dense Character Labeling and Statistical Analysis

Once classifiers are defined for a specific animation style (e.g., series), any new video with the same style can be densely annotated using the classifiers on every frame. Some results showing these examples are shown in the supplemental video. Note that these were

Series (credit ©)	Detected proposals	Clusters to name	Number of characters	Clusters per character	Boxes per character	Relevant Purity[%]	General Purity[%]
1. Bob the builder (HIT)	2,648	20	10	1.7	26.3	95.5	95.5
2. Fairly odd parents (Nickelodeon)	4,215	17	18	1.0	44.3	99.1	99.1
3. Fireman Sam (HIT)	4,633	14	14	1.5	35.4	97.2	79.1
4. Floogals (Jellyfish Pic.)	4,163	4	4	1.0	92.8	99.2	99.2
5. Garfield (Mediatoon)	4,959	13	4	1.4	53.5	99.5	99.5
6. Southpark (Viacom)	5,639	25	21	1.0	16.6	100.0	93.8
7. Land before time (NBCU)	4,795	11	8	1.5	54.5	98.8	98.8

Table 5: *Training videos statistics of seven different animation styles in our* EVALUATION *set that were used for building the classifiers. CAST brings down user effort bounding boxes naming by two orders of magnitude, while still creating a sufficient training-set.*

Series name	Characters [#]	Precision [%]	Recall [%]	Accuracy[%]	F1 [%]	Support [#]
1. Bob the builder	6	97.4	97.4	97.4	97.4	230
2. Fairly odd parents	6	93.8	92.3	92.3	91.9	246
3. Fireman Sam	14	73.1	66.6	70.1	66.7	490
4. Floogals	4	94.7	92.6	93.9	92.6	163
5. Garfield	4	93.7	87.3	87.3	88.0	165
6. Southpark	7	94.5	94.1	94.1	94.2	104
7. Land before time	8	92.2	91.3	91.3	91.3	344
Total	49	91.3	88.8	89.5	88.9	1742

Table 6: *Results on test videos per style of our* EVALUATION *dataset.*

created without temporal coherency for tracking - only running the classifiers on every frame and automatically labeling each character classified.

Using CAST, creative studios and media companies can benefit from a service that can automatically index and expose information both at a video and series level. Labeling each frame in the video with the characters that appear allows to gather important statistics on episodes as well as whole series. This can assist animation productions in data management especially as more and more animation content is being created. For example, gender bias has become an important topic which affects us daily. By knowing the characters' gender and running a dense identification we can tell what is the screen time of each character and indicate, for instance, that the 'Cars 2' movie introduces a gender bias of 78% Male screen time.

6. Discussion

We have presented a method to learn a style-specific semantic representation more suitable for animated content using self-supervision. The self-supervision is based on multi-object tracking and building a dataset of triplets to refine a base mapping to the specific style.

Using clustering in the semantic representation space allows us to automatically build character dictionaries for animation movies, to gather training sets for multi-class classifiers of characters, and eventually to automatically perform dense labeling of characters in animation videos. Such a solution allows users and media companies to analyze, reuse, search, and monetize animation content much more easily.

Limitations and Suggestions for Further Research: One lim-
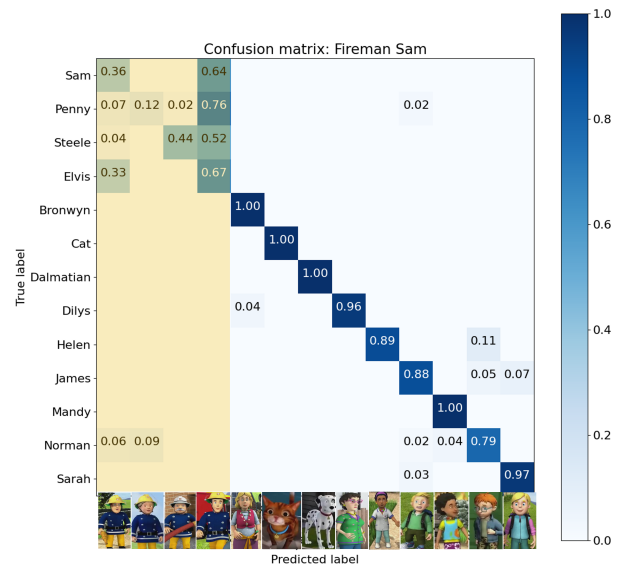

Figure 7: *Limitation in clustering characters that wear uniform. The four firemen cause over 80% of the error.*

itation of CAST is that it relies on the initial detector for proposals. Although we tuned the detector for *recall*, if the detector fails to mark the bounding box of a character as a proposal, the following steps cannot correct this.

An underlying assumption of this research is that different animated characters proposals could be distinguished in the embed-

ding space. However, since the network embedding still depends on visual features, when a subset of characters share common visual features, the embedding may still confuse between them. As a result, the classifiers can also be confused. An example can be seen in 'Fireman Sam' series videos. 14 characters were recognized in this series, while only four of them are responsible for over 80% of the classification error. These characters are the firefighters, and the obvious reason for confusion is their similar uniform (see the confusion matrix in Figure 7).

Our final character labeling did not take advantage of the temporal coherency of the video, but rather treated each frame separately. A more elaborate solution can be defined using smoothing and inference along with tracking.

Style variation in animated content is still extremely large. We refine our basic representation towards specific animation styles using embedding. Another possibility would be to combine such representation with domain adaptation techniques for specialized types of animation (see e.g., Tsubota et al. [TOYA18] on manga comics).

In terms of building the classifiers, our evaluation tested only the two alternatives of class-per-character and class-per-cluster for building the training set. There is a whole range of possibilities between these two extremes that can combine clusters based on some cluster similarity measures to create the training set for characters.

Acknowledgements

References

[AA00] ANDERSEN E. D., ANDERSEN K. D.: The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*. Springer, 2000, pp. 197–232. 5

[ACF*16] ANEJA D., COLBURN A., FAIGIN G., SHAPIRO L., MONES B.: Modeling stylized character expressions via deep learning. In *Asian conference on computer vision* (2016), Springer, pp. 136–153. 2

[ACT16] ALJUNDI R., CHAKRAVARTY P., TUYTELAARS T.: Who's that actor? automatic labelling of actors in tv series starting from imdb images. In *Proceedings of the Asian Conference on Computer Vision - ACCV* (2016), Lai S.-H., Lepetit V., Nishino K., Sato Y., (Eds.), pp. 467–483. 2

[AF86] ATALLAH M. J., FREDERICKSON G. N.: A note on finding a maximum empty rectangle. *Discrete Applied Mathematics 13*, 1 (1986), 87–91. 7

[AWS*18] AZAB M., WANG M., SMITH M., KOJIMA N., DENG J., MIHALCEA R.: Speaker naming in movies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2018), pp. 2206–2216. 2

[BK73] BRON C., KERBOSCH J.: Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM 16*, 9 (1973), 575–576. 6

[BRPM16] BALNTAS V., RIBA E., PONSA D., MIKOLAJCZYK K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc* (2016), vol. 1, p. 3. 4

[CK08] CAZALS F., KARANDE C.: A note on the problem of reporting maximal cliques. *Theoretical Computer Science 407*, 1 (2008), 564–568. doi:https://doi.org/10.1016/j.tcs.2008.05.010. 6

[CL17] CHU W.-T., LI W.-W.: Manga facenet: Face detection in manga based on deep neural network. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (2017), pp. 412–415. 2

[CZD*19] CHEN Z., ZHANG W., DENG B., XIE H., GU X.: Name-face association with web facial image supervision. *Multimedia Systems 25* (2019), 1—20. 2

[DDS*09] DENG J., DONG W., SOCHER R., LI L.-J., LI K., FEI-FEI L.: Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255. 4

[FCH*19] FU D. Y., CRICHTON W., HONG J., YAO X., ZHANG H., TRUONG A., NARAYAN A., AGRAWALA M., RÉ C., FATAHALIAN K.: Rekall: Specifying video events using compositions of spatiotemporal labels. *arXiv preprint arXiv:1910.02993* (2019). 3

[FOY*16] FUJIMOTO A., OGAWA T., YAMAMOTO K., MATSUI Y., YAMASAKI T., AIZAWA K.: Manga109 dataset and creation of metadata. In *Proceedings of the 1st international workshop on comics analysis, processing and understanding* (2016), pp. 1–5. 2

[FTT15] FERNANDO B., TOMMASI T., TUYTELAARS T.: Location recognition over large time lags. *Computer Vision and Image Understanding 139* (2015), 21–28. 2

[HLS*17] HUO J., LI W., SHI Y., GAO Y., YIN H.: Webcaricature: a benchmark for caricature recognition. *arXiv preprint arXiv:1703.03230* (2017). 2

[HLZ04] HUA X.-S., LU L., ZHANG H.-J.: Optimization-based automated home video editing system. *IEEE Transactions on circuits and systems for video technology 14*, 5 (2004), 572–583. 3

[HSS18] HU J., SHEN L., SUN G.: Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7132–7141. 4, 7

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), IEEE, pp. 770–778. 2, 4

[JAA18] JHA S., AGARWAL N., AGARWAL S.: Bringing cartoons to life: Towards improved cartoon face detection and recognition systems. *arXiv preprint arXiv:1804.01753* (2018). 2

[JLNZ17] JASSIN O., LEVI A., NIR O., ZIV O.: Video segmentation and searching by segmentation dimensions, Apr 2017. 10

[KLS*20] KIM H., LEE E.-C., SEO Y., IM D., LEE I.-K.: Character detection in animation movies using multi-style adaptation and visual attention. *IEEE Transactions on Multimedia* (2020). 3, 4, 6

[LH19] LOSHCHILOV I., HUTTER F.: Decoupled weight decay regularization. In *International Conference on Learning Representations* (2019). 5

[MRS08] MANNING C. D., RAGHAVAN P., SCHÜTZE H.: *Introduction to information retrieval*. Cambridge university press, 2008. 5, 6

[NRB17] NGUYEN N.-V., RIGAUD C., BURIE J.-C.: Comic characters detection using deep learning. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)* (2017), vol. 3, IEEE, pp. 41–46. 2

[NRJ*18] NIR O., RONEN R., JASSIN O., GADA M. M., PIPEK M. G.: Training set sufficiency for image analysis, May 2018. 10

[NZB*20a] NIR O., ZONTAK M., BURNS T. C., SINGHAL A., ZHANG L., LEVI A., SABO H., BAR-MENACHEM I., AMI E., BENTOV E.,

ZAMAN A.: Negative sampling algorithm for enhanced image classification, Apr 2020. 10

[NZB*20b] NIR O., ZONTAK M., BURNS T. C., SINGHAL A., ZHANG L., LEVI A., SABO H., BAR-MENACHEM I., AMI E., BENTOV E., ZAMAN A.: Semi supervised animated character recognition in video, Apr 2020. 10

[OON*18] OGAWA T., OTSUBO A., NARITA R., MATSUI Y., YAMASAKI T., AIZAWA K.: Object detection for comics using manga109 annotations. *arXiv preprint arXiv:1803.08670* (2018). 2

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in pytorch. In *NIPS-W* (2017). 5

[Phi07] PHILBIN J.: Oxford buildings dataset. *http://www. robots. ox. ac. uk/˜ vgg/data/oxbuildings/* (2007). 2

[PRNZ*11] PEREIRA D. A., RIBEIRO-NETO B., ZIVIANI N., LAENDER A. H., GONÇALVES M. A.: A generic web-based entity resolution framework. *Journal of the American Society for Information Science and Technology 62*, 5 (2011), 919–932. 6

[RF17] REDMON J., FARHADI A.: Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7263–7271. 2, 4

[RHGS15] REN S., HE K., GIRSHICK R., SUN J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015), pp. 91–99. 4

[SEA19] SHEN X., EFROS A. A., AUBRY M.: Discovering visual patterns in art collections with spatially-consistent feature learning. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 9278–9287. 2, 4

[SHN21] SOMANDEPALLI K., HEBBAR R., NARAYANAN S.: Robust character labeling in movie videos: Data resources and self-supervised feature adaptation. *IEEE Transactions on Multimedia* (2021). 2

[SKGN18] SOMANDEPALLI K., KUMAR N., GUHA T., NARAYANAN S. S.: Unsupervised discovery of character dictionaries in animation movies. *IEEE Transactions on Multimedia 20*, 3 (2018), 539–551. 2, 3, 6, 7

[SKJ*19] SOMANDEPALLI K., KUMAR N., JATI A., GEORGIOU P. G., NARAYANAN S.: Multiview shared subspace learning across speakers and speech commands. In *INTERSPEECH* (2019), pp. 2320–2324. 2

[SKP15] SCHROFF F., KALENICHENKO D., PHILBIN J.: Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 815–823. 2, 4

[Sta18] STARK L.: Facial recognition, emotion and race in animated social media. *First Monday 23*, 9 (2018). 2

[STSS19] SHARMA V., TAPASWI M., SARFRAZ M. S., STIEFELHAGEN R.: Self-supervised learning of face representations for video face clustering. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)* (2019), IEEE, pp. 1–8. 2

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (2015). 4

[TOYA18] TSUBOTA K., OGAWA T., YAMASAKI T., AIZAWA K.: Adaptation of manga face representation for accurate clustering. In *SIGGRAPH Asia 2018 Posters*. Association for Computing Machinery New York NY United States, 2018, pp. 1–2. 2, 10

[WCGV19] WANG X., CAI Z., GAO D., VASCONCELOS N.: Towards universal object detection by domain attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 7289–7298. 3

[WWW*19] WANG C., WANG Y., WANG Y., WU C.-T., YU G.: mussp: Efficient min-cost flow algorithm for multi-object tracking. *Advances in Neural Information Processing Systems 32* (2019), 425–434. 5

[WZM12] WANG X.-J., ZHANG L., MA W.-Y.: Duplicate-search-based image annotation using web-scale data. *Proceedings of the IEEE 100*, 9 (2012), 2705–2721. 6

[YNS19] YANIV J., NEWMAN Y., SHAMIR A.: The face of art: landmark detection and geometric style in portraits. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 1–15. 2

[YS11] YU J., SEAH H.-S.: Fuzzy diffusion distance learning for cartoon similarity estimation. *Journal of computer science and technology 26*, 2 (2011), 203–216. 2

[YZL19] YANG Z., ZHANG Y., LUO J.: Human-centered emotion recognition in animated gifs. In *2019 IEEE International Conference on Multimedia and Expo (ICME)* (2019), IEEE, pp. 1090–1095. 2

[ZGW16] ZHANG S., GONG Y., WANG J.: Deep metric learning with improved triplet loss for face clustering in videos. In *Pacific Rim Conference on Multimedia* (2016), Springer, pp. 497–508. 2

[ZHEL*13] ZHANG T., HAN Q., EL-LATIF A. A. A., BAI X., NIU X.: 2-d cartoon character detection based on scalable-shape context and hough voting. *Inf Technol J 12*, 12 (2013), 2342–2349. 2

[ZLLT16] ZHANG Z., LUO P., LOY C. C., TANG X.: Joint face representation adaptation and clustering in videos. In *European conference on computer vision* (2016), Springer, pp. 236–251. 2

[ZLN08] ZHANG L., LI Y., NEVATIA R.: Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (2008), IEEE, pp. 1–8. 5

[ZWS11] ZHU C., WEN F., SUN J.: A rank-order distance based clustering algorithm for face tagging. In *CVPR 2011* (2011), IEEE, pp. 481–488. 2

[ZXLH09] ZHANG Y., XU C., LU H., HUANG Y.: Character identification in feature-length films using global face-name matching. *IEEE Transactions on Multimedia 11*, 7 (2009), 1276–1288. 2

[ZZR*20] ZHENG Y., ZHAO Y., REN M., YAN H., LU X., LIU J., LI J.: Cartoon face recognition: A benchmark dataset. In *Proceedings of the 28th ACM International Conference on Multimedia* (2020), pp. 2264–2272. 2