Multi-Perspective Action Recognition Through a Contrastive Objective

Chris Blythe
University of Central Florida
4000 Central Florida Blvd, Orlando, FL 32816
cblythe@knights.ucf.edu

Abstract

This paper proposes a novel extension of previous methods for action recognition from video clips, invariant of changes in perspective. We attempt to achieve this goal by classifying actions using spatiotemporal feature embeddings generated by a 3-dimensional convolution network, trained to extract perspective invariant features. This invariance is learned during training through a triplet loss objective targeting samples from multi-perspective videos. Through this objective, related features from two separate views of the same action can be brought together into the same feature space, while features from separate actions in the original view are pushed further away. Quantitative results are generated by training and testing our model and a comparable C3D model on actions from the IXMAS dataset.

1. Introduction

With the sudden surge in computational power afforded by modern hardware, the capabilities of deep learning networks have seen a dramatic increase. This increase has led to numerous concurrent innovations within the field of computer vision, which in in turn influence and extend each other to form new discoveries. At the same time, visual content has become increasingly more available and varying in form and quality. Previous research methods would primarily focus on recognizing actions from conventional RGB video. However, today we have access to numerous forms of content, such as: 360° videos, 3d generated content, and satellite images [1, 2]. It is common for these differing forms of content to be used as providers of perspective shifts or invariance [2, 1, 3]. In other words, many methods use this content to provide or minimize a perspective (i.e. aerial perspective to ground, multi-view to single) to target content of a differing form. In this paper we intend to present this concept as a possible method for improving perspective invariance regarding human action recognition.

Human action recognition has always been at the forefront of computer vision research due to its numerous research implications and applications, including: video analysis, surveillance, and human-computer interaction [1, 4, 5]. Actions can be thought of as simple motions created by a human body that cannot be broken down into further primitives [1]. Action recognition is the act of correctly classifying a given video clip as representative of a discrete class (i.e. running, jumping, etc.). Action detection is a related concept, but differs in that it deals with localizing the action along the space and time of the clip. We can assign classifications to video clips by using deep learning networks to extract various features. These features are encoded with properties of the video, such as changes over time and space, that relate to the classification task. We can then process these features using a classifier that assigns a probability distribution over each possible action that the clip could represent. Networks like Convolutional 3D (C3D) [4] allow for the efficient extraction of relevant features over the width and height of the video, but also the temporal dimension of all the frames in the video. In this manner we can break down a clip into spatiotemporal features that inform our decision of which action is taking place.

The main contribution of our work is a novel extension of previous work done with perspective dynamics in feature vectors [3] and 3d convolutional networks [4] to achieve classification of actions in video clips. We accomplish this by using multi-perspective content to train an action contrastive objective for perspective invariance as part of a standard C3D architecture. This allows us to generate perspective invariant feature embeddings, trained to distance disparate classes, that can provide us with better action recognition results. Through our experiments we quantitatively attempt to demonstrate that this invariance can lead to better action classification scores than a standard C3D model.

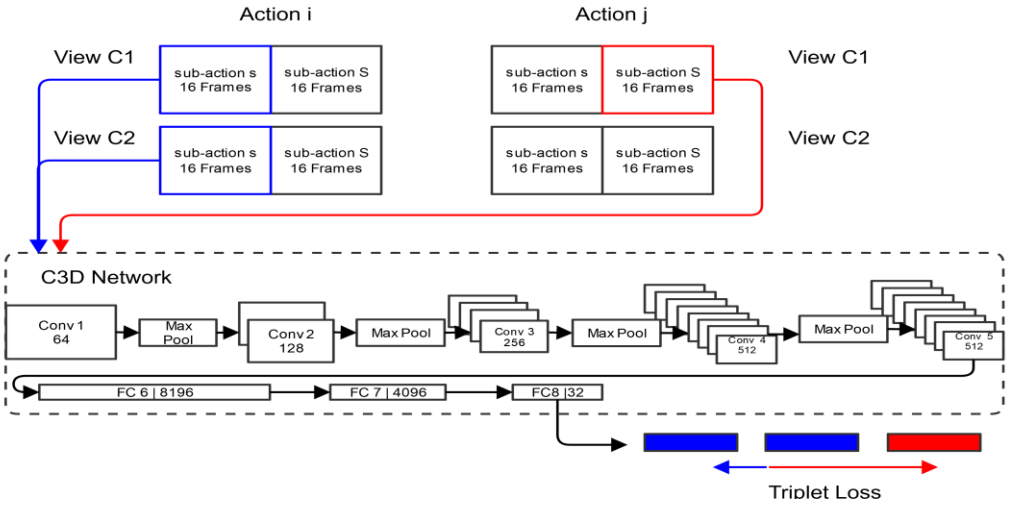

Figure 1: C3DTCN architecture uses two disparate actions to provide our triplet loss samples. The anchor and positive samples come from the same action clip (i), but different views (c1,c2). The negative sample is taken from a different action (j), but the same view(c1). Each sample is passed through the C3D network to create a feature vector that informs our triplet loss. The loss brings the anchor and positive closer together, while pushing away the negative feature vectors.

2. Related Works

Using different perspectives during the training phase to minimize or augment target properties within a feature space, is a fairly common method that can extend to many possible use cases [2, 1, 3, 6]. This perspective shift can be related to camera angle [2], or even a stylistic change in representation [6]. Our approach is most closely related to the work done by Sermaner et al. on Time Contrastive Networks [3]. The goal is to use a contrastive triplet loss over three related samples.

Three sub samples are taken from each clip: one anchor and negative clip from the same perspective, but different time points, and one positive clip from the same time as the anchor, but different perspectives. They then use feature embeddings of all three clips, and a loss targeted at bringing the anchor and positive features into the same space, while pushing the negative features further away [Figure 2]. The authors used these distant time slices as negatives to enforce and recover the smallest orientation changes between frames for their imitation objective. Our method differs in that our negative sample does not target a different slice of time in the same clip, but rather a completely different action. This is done to ensure that we do not negatively sample parts of a looping action (i.e. walking) and that we optimize for classification by pushing away features from unrelated actions. Also, our method differs in that we only employ the TCN loss objective and use a C3D architecture to retrieve the spatiotemporal feature embeddings for our triplet samples.

The C3D network created by Tran et al., provides a deep 3-dimensional convolution architecture that accurately models spatial and temporal changes as feature vectors [4]. C3D is able to accomplish this through the smart use of 3d conv and 3d pooling layers that process samples spatially, by height and width, but also temporally by frame. Intuitively, this architecture allows the network to first focus on the appearance of objects, before tracking related motions in continuing frames [4]. Since C3D is designed to handle computationally expensive content like video clips, this architecture is engineered to be “efficient, compact, and easy to use” [4].

Figure 2: Triplet loss used to enforce relationship between anchor and positive sample, while distancing the negative sample. **[7]**

To achieve this, Tran et al., have provided optimal hyperparameters like: image size, frame depth, kernel depths, etc. For our model we used the general C3D architecture provided, along with the advised: 16 frames per sample and spatial dimensions of 112 by 112. Our purpose for this network differs from Tran et al.’s in that we are not directly using C3D for classification. We are strictly using C3D to create feature embeddings [Figure 1] that we train using our action contrastive loss. Once fully trained, the resulting feature embeddings can be used by any future network. In our case we apply the generated features to a simple classification network.

3. Method

We approach human action recognition as a problem of optimal feature extraction for a feature embedding that is invariant to perspective and conditioned on separate actions. Once trained the C3D network with action contrastive loss should be able to produce portable feature embeddings useful for most action recognition networks

Concretely, our contrastive network attempts to enforce the following relationship. For any given sample x over actions i, j and camera perspectives $c1, c2$ we create combinations of samples x from our dataset X. The anchor, and positive samples are taken from the same action (*i*) but different camera perspectives (*c1*,*c2*). The negative (*n)* is taken from the same camera perspective (*c1*), but a different action (*j*). Our loss attempts to ensure that the positive feature vector p is kept close to the feature space of anchor a, while increasing the distance of negative features n. The minimum distance in this relationship is enforced by the margin value α.

$$dist(a-p) + \alpha - dist(a-n) \leq 0$$

$$where \ \ a,p,n \in \tau \ all \ combinations$$
$$a = f(x_{c1}^{i}), \qquad p = f(x_{c2}^{i}), \qquad n = f(x_{c1}^{j})$$

The goal being, that as our training progresses our average loss would diminish along with the average distance of each positive to anchor sample. Validation of the training performance would be shown as the percentage of samples with positive distances near or at 0 per epoch. In this manner multi perspective features from the positive sample of one action are brought together into the same feature space as the anchor, while foreign features from a different action and time are pushed further away. This method should encourage the benefits of TCN loss, "invariance to viewpoint, scale, occlusion, motion-blur, lighting and background" [3], as well as possible separations between trained actions within the feature space.

4. Implementation

Our full action recognition implementation is made up of two parts: A C3D model conditioned on our contrastive loss [Figure 1], and a simple multilayer perceptron functioning as our classification network. These networks are implemented using python and the pytorch framework.

The C3D model is pretrained using the UCF Sports dataset to allow for faster convergence. We begin to train the model using a custom pytorch dataset for the multi-view IXMAS dataset [
Figure **3**]. This is a simplistic dataset consisting of individual frames with limited background noise, 11 different actors, 15 rudimentary actions, and 5 fixed camera angles. Each of the five views are synchronized for the duration of each action. Initial analysis of the dataset showed an imbalance to the action distribution of the training data (specifically for the 'nothing' action). As a preprocessing measure the training dataset was normalized so no action has more than 10% representation. We could have also addressed this issue by biasing the weights for that class in the model.

Figure 3: Examples of IXMAS dataset actions and camera angles **[5]**.

For training we employed 79 actions from 6 different actors, with 20 combinations of viewing angles, leading to a total of 1580 training samples. For validating our training per epoch, we used a separate subsection of IXMAS data consisting of 200 samples over 10 actions. Each sample is originally sized at 390 by 291 pixels. Due to the simplicity of the dataset, and the localization of each action, we simply center crop the sample to an even 224 by 224. We then down-sample each sample to the C3D optimized 112 by 112. The positive and anchor sample are pre-arranged as sets when the dataset is created. Each sample consists of 16 frames from a random starting point within the clip, recalculated whenever the sample is retrieved. The anchor and positive samples are guaranteed to start and end at the same points. The negative clip to the anchor is provided by a completely different action chosen randomly at retrieval, but with the same actor and view. The resulting samples within a batch of size 16, will consist of 3 channels, 16 frames, and 112 pixels by 112 pixels.

Each sample is passed on to the standard C3D network to generate feature vectors $a,\ p,\ n,$ of size 32 for each entry in a batch. 32 is chosen as it is a compact feature representation and fits with the findings of Semaner et al.'s TCN implementation [3]. The action contrastive loss is then calculated and backpropagated to minimize the

distance between future anchor and positive samples. We use Stochastic Gradient Descent (SGD) as our optimization function with stepped learning rate degradation scheduled at 50, 100, and 200 epochs. SGD was chosen over methods like ADAM as it has been shown to adequately generalize and avoid overfitting in diverse conditions [8]. At the time of this writing we were only able to train up to 100 epochs. Through experimentation we were able to identify $1.0e^{-7}$ as the ideal starting learning rate for our implementation.

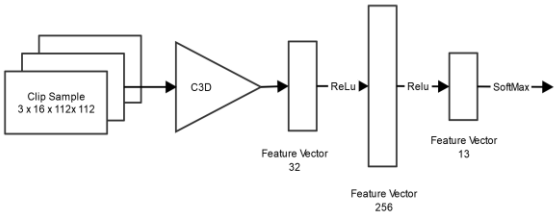

Figure 5: Classification network using features generated with C3D network trained on contrastive loss.

The C3D model itself is a standard configuration consisting of 5 3-dimensional convolutional layers, broken up by 3d-max pooling to reduce the dimensions to manageable features. Three Fully convolutional layers (of size 8192, 4096, and 32) are used to form our output feature vector. The last fully connected layer does differ from the C3D standard, but we have decided on a compact 32 value output vector as depicted in the original TCN paper [3]. Though small, this vector should contain the most important features, extracted from both the anchor and positive samples, within the same embedding.

After our network is trained we apply the resulting feature embedding to training an action classifier network [Figure 5] on a separate set of IXMAS actions. In this network, we only retrieve one sample at a time, and pass it to our trained C3D network to generate a feature vector. This feature vector is passed on to a simple multilayer perceptron containing 3 fully connected layers (of sizes 32, 256, and 15) that culminate with a SoftMax activation for classification. Two dropout layers are added between the fully connected layers to prevent overfitting by ensuring that no neuron interdependencies are formed. The end SoftMax layer then produces a probability distribution over the 15 possible actions that this sample could represent. A simple cross entropy loss, based off the SoftMax prediction results, is then backpropagated through the network.

5. Experiments

To properly train our models various combinations of hyperparameters were tested. Two methods were used for measuring performance on the C3D model. Average distance loss by each epoch, and validation loss against the

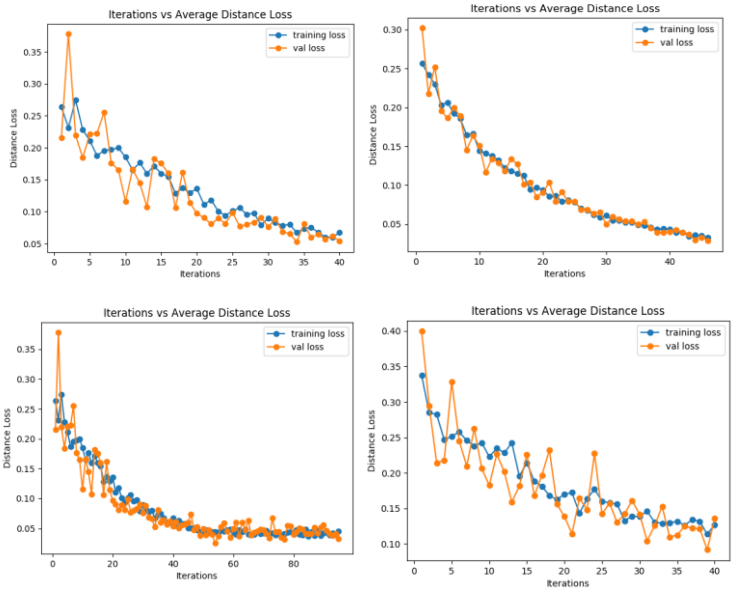

Figure 4: Average distance loss at batch size 2 (top-left) and batch size 6 (top-right). Final training results for 100 epochs (bottom-left). Margin value test at 0.2 (bottom-right).

validation set. Our expectations were to see a continuously diminishing average loss per epoch until convergence as close to 0 as possible. We also expected to see a consistently diminishing validation loss as more positive distances come closer to 0.

During training of the C3D model we initially attempted a lower batch size of 2 due to GPU memory limitations. This lead to seemingly erratic distance loss and validation results during our training. Over the epochs the changes subsided, but the initial training always contained erratic jumps. Later, we applied the network on a slightly more powerful computer to test a larger batch size of 6 and noticed smoother results [Figure 4]. This is due to the larger batch size providing a more consistent mean loss over 6 samples rather than the initial 2. Essentially, a larger batch can minimize the potential impact of any independent outliers. Sadly, a larger batch size also means more memory usage by the memory limited GPU. Without an increase in computational space, this situation results in a trade off where network depth, image resolution, or batch size must be reconsidered in order to achieve better results.

Another issue that arose in training was the correct value for the margin in our loss. Considering the small localized changes in the IXMAS dataset, the actual difference between our different samples may be quite small. A standard value of 2.0 was originally used, but the validation loss would be static at 1.0 and values did not converge on a low enough average distance loss. The idea being that pushing the negative distance too far away

would create an imbalance in our results. The TCN implementation provided by Sermaner et al. uses the margin value of 0.2. But for our dataset and model implementation that value only resulted in average results [Figure 4]. After studying the average distance between the positive and negative samples we decided on a margin value of 0.01to provide acceptable results. However, our average distance loss never successfully diminished past the margin point, hovering roughly around value 0.02l9 [Figure 4]. It's possible that given more computation time the loss could have converged on smaller values, but that will be left for future work.

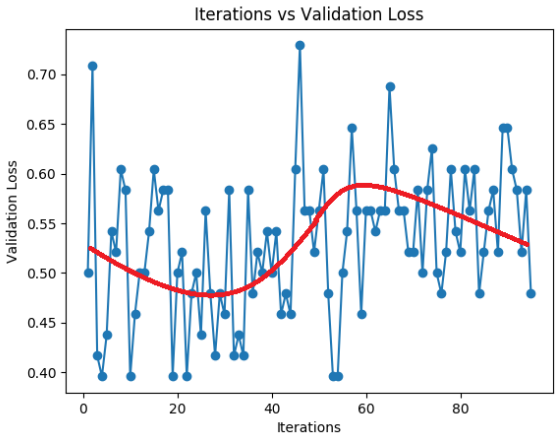

Figure 7: Validation loss results for C3D training.

Our training also resulted in some consistently strange validation results, regardless of hyper parameter changes. Often the validation loss would oscillate wildly with each new epoch [Figure 7]. This could be explained by the innate variations of selecting random samples, frames, and negative actions. However, such results should have been relieved by a larger batch size that averages out any large outliers. Also, our initially low learning rate should not have allowed for such wild changes as a result of our loss function. However, the general trend of the validation results does seem to be consistent with our stepped changes of learning rate (50, and 100).

Once we had a trained C3D network and feature embedding we began to test it's benefits on a simple classification network. Training for the classification network used a separate subset of the IXMAS dataset as a validation step. The goal being that our feature embeddings should be generic enough to benefit any other dataset, even if the resulting data has never been trained on. To further validate our theories, we trained and tested a separate version of our classification network with a standard C3D model.

During training the standard C3D classifier showed some erratic behavior in it's validation loss. While our implementation kept a somewhat consistent decrease in

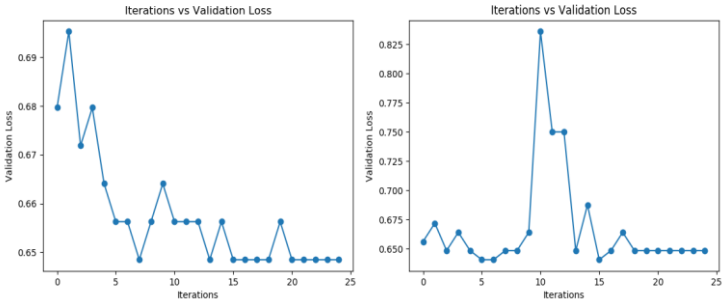

Figure 6: Classification training results for our method (left) and a standard c3d model (right).

validation loss before flattening out [Figure 7]. This invariance to such drastic change informs a sense of structure within our model's feature embedding that does not show in the standard C3D implementation. However, both methods do show a discrete limitation in their learning when they simultaneously stop their validation loss decrease around epochs 10 and 15.

In testing both methods performed at nearly identical levels. With the standard model providing 34.89% accuracy and the contrastive objective model model providing 35.67% accuracy. Objectively these are poor scores for classification, but with further training and hyper parameter changes we could see better results. These results may not seem like much of an improvement, but they at least provide the assurance that our learned feature embedding is both generic and transferable.

6. Future Work

We have confirmed that our perspective invariant feature embeddings can be carried over to other tasks such as classification without negative results. Further verification of this concept could be done by applying our method trained on the IXMAS dataset to other datasets like UCF Sports. The feature embeddings learned through this method should be generic enough that they still could help inform the network of the appropriate actions; Even though our network has never been trained on this data. This opens the door to applying such embeddings to other forms of content aside from video clips.

7. Conclusion

Though our results are inconclusive, we have shown that multi-perspective samples can be used to generate generic feature embeddings that at least do not negatively impact our results. These feature embeddings did also show more structure and stability during training, which may translate into better performance with continuing training.

8. References

[1] S. M. Knag and W. R. P, "Review of Action Recognition and Detection Methods," *arxiv.org*, 2016.

[2] M. Zhai, Z. Bessinger, S. Workman and N. Jacobs, "Predicting Ground-Level Scene Layout from Aerial Imagery," *arxiv.org.*

[3] P. Sermanet, C. Lynch, Y. Chebotar, H. Jasmine, E. Jang, S. Schaal and S. Levine, "Time-Contrastive Networks: Self-Supervised Learning from Video," *arxiv.org.*

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *arxiv.org*, 2015.

[5] J. Liu and M. Shah, "Learning Human Actions via Information Maximization".

[6] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," *arxiv.org.*

[7] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," *arxiv.org.*

[8] I. G. P. b. S. f. A. t. SGD, "Shirish, Nitish, Keskar; Socher, Richard," *arxiv.org*, 2017.