Learning Spatio-Temporal Classification and Localization using Consistency-based Semi-supervision on UCF-101 Dataset

Abraham Jose
CAP 6412: Advanced Computer Vision
University of Central Florida
abraham@knights.ucf.edu

Abstract

Most of the detection and classification tasks are based on images and compared to that very less research is being done in the video classification and detection domain. This is to some part because of the lack of enough annotated video dataset available to public as the cost incurred to create a video dataset annotated frame by frame for bounding box is huge. Inorder to tackle this problem we can use semi-supervised learning technique, for example, ***Consistency-based Semi-supervised Learning for Object Detection****. In order to test this self-supervised consistency based learning(both on classification and localization), we are using one of the public dataset available, UCF-101. The dataset offers 101 classes of video dataset, of which 24 classes are annotated with bounding box for localization(detection). The model being used is an extended version of 3D Resnet pre-trained on UCF-101 dataset. The localization is limited to one detection per clip formatted same as UCF-101 annotations.*

1. Introduction

Computer Vision as a research domain have seen many rapid developments and research directions since the boom of the deeplearning and huge datasets available for training the deep-learning models using convolutional neural networks. However, most of the work in the domain are confined to the image processing category where a single image is being used for processing and understanding the data. Hence the work in semi-supervised learning in the domain is also limited to image content. Through this work, semi-supervised learning on activity recognition and localization is targeted using a previous works in the field of semi-supervision using consistency loss in the image detection and classification segment. This work will hopefully be able to direct work in the field of semi-supervised video detection and localization using Consistency based loss function.

2. Background and Related work

The method employed is directly taken from two researches, one is Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition/ Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet? from Kensho et. al. which is based on 3D CNNs for action recognition in datasets such as UCF 101, ActivityNet, Kinetics and HMDB-51. In this specific work, the author examines various CNN architectures by using the spatio-temporal 3D convolutional kernels on available video sets. The research examines flavours of ResNet such as ResNet-200, ResNet-152, ResNet-101, ResNet-51, ResNet-34, ResNet-18 and other variations such as ResNext, DenseNet, Wide ResNet etc.. From the work, my algorithmic choice was ResNet-34 for the feature extractor as the ResNet-18 model is prone to over-fitting in the UCF-101 dataset as described in the paper **[2]**.

The other work is called Consistency-based Semi-supervised Learning for Object Detection from Jisoo Jeong et. al. This particular work is for improving the result in a sparsely annotated dataset for detection purpose. This can be implemented in single-stage and two-stage detectors where the input image will be flipped and will be used for the getting the activation from the model and calculates consistency loss for an image-flipped_image pair. This error will be back propagated to the network along with localization and classification losses. The authors also proposed Background Elimination to mitigate the effects of the predominant background classes in the image which will affect the detection performances. The single-stage detector approach is being directly borrowed in this study.

2.1. Video Dataset

The dataset used is UCF-101, which is one of the most successful dataset available in the action recognition domain in the field. The dataset gained popularity in the field and still used as a popular benchmarking dataset for

the training. The dataset covers 101 datasets for action classification which includes dataset covering a wide range of human action classes which are trimmed. The dataset consists of over 13000 trimmed videos and 27 hours of annotated dataset. Also the dataset has 24 classes which are annotations for the bounding boxes of the corresponding actions. The dataset allows us to be flexible to be able to use the 77 classes which is not annotated and use them to generate the unsupervised detection technique and utilize the consistency loss as described in the **[1]**.

The bounding box annotations for dataset are available for 24 classes and of these 24 classes, there are atleast 100 annotated videos per class and there are atmost 163 annotated videos per class with an average of 130 annotated videos per class. Of the available annotated video sets there are only few frames which are annotated per class and on an average there are 139 frames annotated per video. The rest of the frames are not annotated and the rest of the video is considered to be the unannotated dataset and is considered as a classification sample as opposed to classification and detection(bounding box) sample in case of annotated samples in the video.

Table 1: 24 Classes with ground truth bounding box annotation.

Action	No of Annotations	No of Videos
Basketball	5142	134
BasketballDunk	4904	131
Biking	27958	134
CliffDiving	877	138
CricketBowling	5424	139
Diving	16983	149
Fencing	14031	111
FloorGymnastics	18899	125
GolfSwing	17116	139
HorseRiding	31143	164
IceDancing	38892	158
LongJump	16842	130
PoleVault	22443	149
RopeClimbing	19556	119
SalsaSpin	15307	133
SkateBoarding	16321	120
Skiing	29145	135
Skijet	22356	100
SoccerJuggling	42223	147
Surfing	17833	126
TennisSwing	7158	163
TrampolineJumping	18419	115
VolleyballSpiking	3529	112
WalkingWithDog	25974	123

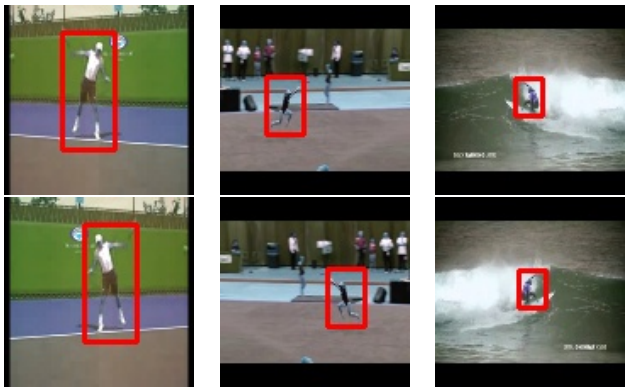
Table 1. provides the class-wise distribution of annotated frames available in the dataset. These annotations are assumed to annotate human centered actions and hence we consider that the annotations tightly bounds the human which is not the case in some cases.

3. Method

3.1. Data Pre-Processing

As described in the previous section we have two sets of the data available in UCF-101 dataset that can be used to train the model both in classification and detection tasks. A pytorch datasloader is written to extract the data from the pickle file and the video which is attached to the code available at utils/data_loader_error.py. The dataloader is designed to create clips of image size 128x128x3 and of length 16 frames from the a video in the videoset, available seperately for both training and testing. The dataloader is designed in such a way that the dataloader will randomly pick an index of the video from range of length of available video set and then uses the index to load the corresponding video and randomly initialise the starting and ending frame number(start frame + length of the clip). If the corresponding loaded frame has the bounding box annotations available in the pickle file, then they are loaded to a tensor shaped 16x4 that holds annotations initialized with 0s. If there are no available annotations in the selected frames, it will return initialized tensor of zeros.

Alongside with the dataset class, inorder to flip the clips and the bounding box values, the function, HorizontalFlip is added to the videotransforms.py script. The script is used for flipping the labels and to flip the co-ordinates of the bounding boxes. Samples of the flipped images and corresponding bounding box is follows. We can see that in class Floor Gymnastics, there are many people in the dataset however, only the action class is annotated, which forces the network to discard features and negatively affects the training process.



(a) Tennis Swing (b) Gymnastics (c) Surfing

Figure 1: Sample images and ground truth from original and flipped clips

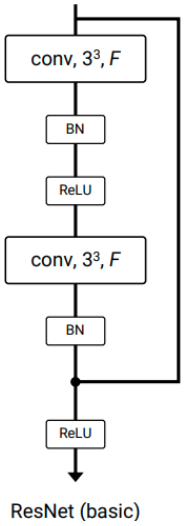


Figure 2: Basic building block used for Resnet-18/34

3.2. Model : 3D Resnet34

The model is inspired by the Resnet-34 architecture and is the version that is inflated with 3D feature kernels in vanilla Resnet-34 architecture with an extra dimension. The input dimension to the model is (-1,3,16,128,128) and the output dimension is (-1,101) and the logits will be extracted using the softmax function.

The basic building block used for Resnet-34 is follows. The basic block has 3x3x3 kernel of F features followed by Batch Normalization, ReLU and the 3x3x3 kernel again followed by Batch Normalization which is followed by skip connection for residuals within the block followed by another ReLU inorder to contain the loss bounded and to ensure smooth loss propagation through the network.

The full architecture of 3D Resnet-34 is as described in Table 1. The first convolutional layer has 7x7x7 kernel followed by the N number of basic blocks as described above in the figure having F feature kernels. For example, the conv1 layer is followed by 3 basic blocks having 64 kernels each. The last layer of the network has 101 FC layers after the Average pooling layer followed by the FC layer having 101 outputs with Softmax activation.

Table 2: Resnet-34 Architecture

conv1	conv2_x	conv3_x	conv4_x	conv5_x	FC
(7,7,7),64F temporal-1 spatial-2	64F, 3Blocks	128F, 4Blocks	256F, 6Blocks	512F, 3Blocks	101

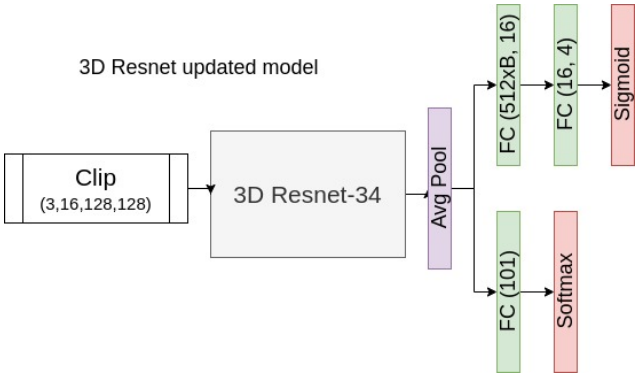


Figure 3: Model 1: with 2 FC layers

The 3D-Resnet-34 architecture has pretrained model available for kinects dataset and the average accuracy on validation set of Kinects using 3D-Resnet-34 is 71% which is close to the performance of 3D-Resnet-50, 72.3% and 3D-Resnet-101 with 73.2% accuracy. hence the pretrained model of 3D-Resnet-34 is the choice of the architecture for the implementation. The 3D-Resnet-34 was trained on the UCF-101 dataset and the pretrained model is being used to extract the 3D CNN features and then use the weight

3.3. Final Model Architecture

The choice of the model for the last model is based on assumption that the feature extracted from a clip will be able to represent the clip's action class and the bounding box of the human doing action in the center frame of the video clip. Based on this assumption, the model was an extension of the classifier network with another set of fully connected layers to regress the bounding box values in the formay (x_left,y_up,w,h). To achieve this, the 3D-Resnet-34 model was attached with FC with following configurations. The FC layer is attached after the average pooling layer following the last layer of the CNN block.The models used for training the model is as described in the Figures 3 and Figure 4.

The first model being used to train the model is having 2 FC layers with 16 and 4 output layers followed by a sigmoid activation function as shown in the Figure 3. The sigmoid activation is to ensure that the model is able to produce results in the range 0-1 which will be correspond to the 4 values of the bounding box in the format mentioned above. The model has been trained with 2 setups. One with frozen feature layers and other with unfrozen feature layers. This ensures us to learn only the FC layers that we attached to the model and not any other layers by using the localization loss. The training procedure will be explained in detail in the section Experiments.
The second model that is used to train the model is as discribed in the Figure 4. The model uses 4 FC layers

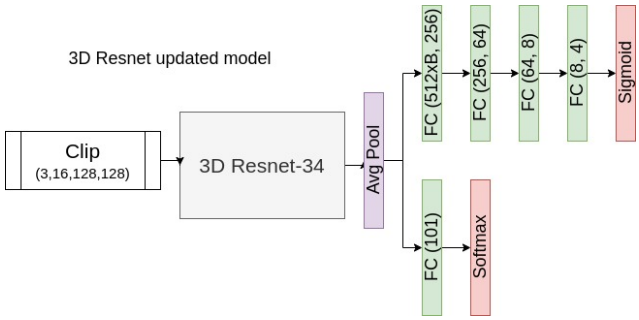


Figure 4: Model 2: with 4 FC layers

inorder to learn bounding box points form the feature layers. This is to make sure that from the features learned from the network, the model has enough flexibility and capacity to learn from the features extracted using the 3D kernels in the 3D Renset-34 network. Following the Average pooling layer, the model has FC layers with 256, 64, 8 and 4 layers respectively followed by Sigmoid activation layer.

During the training process, it is found that the FC layers does not requires activation layers except for the last layer where the output will be bounded in between [0,1]. Hence the FC layers were used without any activation layers inbetween the layers.

4. Experiments

4.1. Training 3D-Resnet-34 on UCF-101

Using the pretrained 3D-Resnet-34 on Kinetcs dataset, the model was trained on the UCF-101 using transfer learning technique by updating the last FC layers in the model from 400 output to 101. Categorical Cross-entropy was used as the loss for the model while fine-tuning the model in UCF-101 dataset.

After 200 epochs of training the model, the accuracy and the loss profile is as follows in Figure 5. The model was able to achieve classification accuracy of 96% on the training set and 80% on the validation set respectively by fine tuning on the Kinetics dataset. For training and testing we used the split-1 for UCF-101 dataset. The following graph shows the training and validation accuracy for the training and validation set in the split-1 of UCF-101 dataset. The model saturates around 50th epoch and we can see that the training and testing accuracy and loss are consistent afterwards. The number of batch used for training the model from the Kinetics dataset was of size 128 and each epoch is having 75 iterations and in total there has been 15000 iterations inorder to train the model from scratch.

We can see that the there is a huge gap between the training

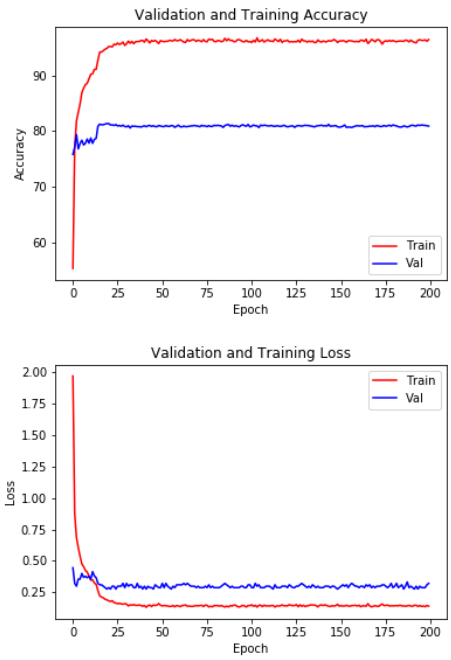


Figure 5: 3D-Reset-34 Training and Validation Accuracy and Loss.

and testing results. The reason is that the model was able to trained on Split 1 in UCF-101 dataset and the features fine-tuned from the Kinects dataset has been favourable to the training set. Hence the model is overfitted on the training dataset as opposed to the testing dataset. That is the reason for a huge performance gap between the accuracy(of 16%) of training and validation set of the UCF-101 split 1. It should be noted that the train and test dataset composes of the data splited in the same category, however distinct videos. This brings our attention to the fact the model is not quiet generalize-able with the current model and the dataset split(split-1 is taken directly from the UCF-101 prescribed dataset split).

4.2. Loss Implementation

Classification Loss: The classification loss employed for training the based 3D Resnet-34 model is CrossEntropy Loss defined by,

$$L_{classif} \textbf{ = H(y,p) = } \sum_i y * log(\frac{1}{p}) = -\sum_i y * log(p)$$

Where y and p are Actual and Predicted classes probabilities after the Softmax output from the last FC layer of the classification network. The Cross Entropy measures the probability distribution of the predicted and actual probabilities in 101 classes. So by minimizing the loss using SGD, we are essentially reducing the distance between the predicted and actual probability distribution of the predicted and the actual

probabilities. For the actual classes, the vector is one hot encoded with rest of the classes as 0 and the actual as 1. The loss is calculated per clip and averaged for the whole mini-batch.

Localization Loss: The localization Loss is defined by the MSE Loss or L2-Norm. Since the localization of the bounding box is limited to 4 data-points per clip(if there exists a ground truth for the central 2 frames to average the bounding box in the clip), the regression should be able to learn it from the features extracted from the clip's action. The loss is employed only if there exists a ground truth and if there is no ground truth in the clip input the loss is not computed. The loss is defined as follows,

$$L_{loc} \textbf{ = MSE(y, p) = } \frac{1}{4}\sum(y-p)^2$$

Similar to classification loss, the localization loss is also used by averaging in a minibatch for multiple inputs. The MSE Loss was the choice for this regression problem because this L2-Norm is capable of delivering a stable unique solution compared L1 Norm such as MAE(Mean Absolute Error).

Consistency Loss: The Consistency loss is based on the the paper **[1]**. The Consistency loss is calculated in semi-supervised fashion, which means that the loss is calculated evenif there exists no ground truth especially for the localization. The consistency loss is calculated based on the responses for a batch of clips and its corresponding flipped clips. Consistency loss is defined by Jensen-Shannon Divergence(**JS**) which is used as consistency regularization loss. Since the architecture is consistent a single stage detector, the proposed single stage consistency loss in the paper **[1]** is demonstrated in Figure 6.

The consistency loss is calculated for all the inputs which is further split into L_{con_loc} and $L_{con\_classif}$. The average is considered over a batch of clips to calculate the classification consistency loss for a batch. If C and $\hat{C}$ are clip and flipped-clip respectively and $f^k(c)$ and $f^{k'}(\hat{c})$ be the model responses, the JS consistency classification loss is defined for the clip is defined by,

$$L_{con\_classif}(f^k_{cls}(c), f^{k'}_{cls}(\hat{c})) \textbf{ = JS}((f^k_{cls}(c), f^{k'}_{cls}(\hat{c}))$$

k and k' are the instances of the output for a batch in this case where there is only one prediction for the class and the bounding box. The localization consistency loss is defined by the change in the localization for the clip c and the flipped-clip $\hat{c}$ the displacements be $[\Delta cx, \Delta cy, \Delta w, \Delta h]$ and $[\Delta c\hat{x}, \Delta c\hat{y}, \Delta \hat{w}, \Delta \hat{h}]$ respectively,

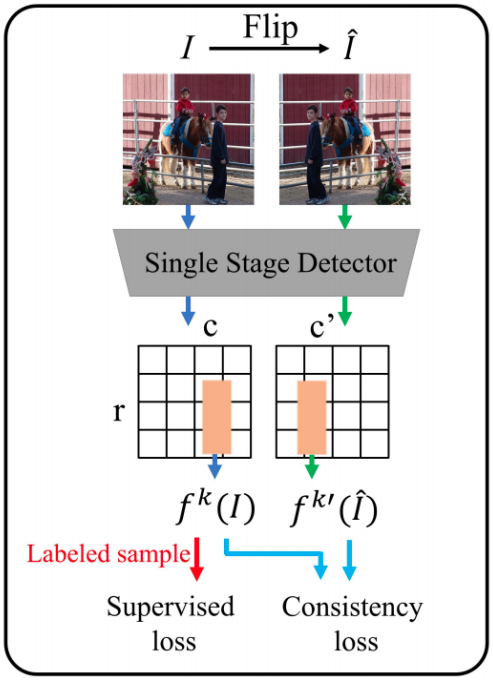


Figure 6: Single Stage Consistency Loss Calculation

$$L_{con\_loc}(f^k_{loc}(c), f^{k'}_{loc}(\hat{c})) \textbf{ = } \frac{1}{4}(\|\Delta cx^k - (\Delta c\hat{x}^{k'})\|^2 + \|\Delta cy^k - \Delta c\hat{y}^{k'}\|^2 + \|\Delta w^k - \Delta \hat{w}^{k'}\|^2 + \|\Delta h^k - \Delta \hat{h}^{k'}\|^2)$$

The average of consistency localization and consistency classification loss is taken over a mini batch and summed up to get the overall consistency loss.

$$L_{con} = L_{con\_classif} + L_{con\_loc}$$

The total loss is the sum of consistency loss, localization loss and the classification loss. A weight value $w(t)$ is used based on the epoch to ramp the consistency loss which will help to deliver a stable training profile. The weight ramping can be ramp-up or ramp-down and here ramp-up is being employed as the consistency loss is significantly small when training is started and it slowly and steadily grows as the localization and classification loss saturates.

$$L_{total} \textbf{ = } L_{classif} \textbf{ + } L_{loc} \textbf{ + } w(t) * L_{con}$$

4.3. Training 3D-Resnet-34

Inorder to train the model with best results possible, the model was loaded and the weights were copied from trained model to the model described in Figure 3 and Figure 4 using the script described in the **models/res_34.py** file in the

submission. The feature weights are taken from the trained model and then weights from the model trained model(3D Resnet-34) was copied to the final architectures Model-1 and Model-2. The following training procedures were used to ensure that the model was able to learn the bounding box values from the features.

1) Copy feature weights to Model-1/2 | Train entire model: The weights are being copied from the 3D Resnet-34 model to the new model architectures described in the Figure 3 and Figure 4. The model is allowed to retrain the whole model using the localization, classification and consistency loss as described in the previous subsection after ramping the consistency weight. The parameter of the neural network that determines the training of a certain layer is set using **require_grad** flag and is set to **True**. The model is trained for over 100 iterations where the training saturates and the IoU

2) Copy feature weights to Model-1/2 | Freeze the layers: Inorder to ensure that the model is able to learn localization from the learned features, the weights are updated in the new models as described in the Figure 3 and 4 and then the FC layers for the localization was trained for 15 epochs initially to see the improvement in the performance in the localization capability of the network followed by 100 epochs described in the previous first protocol. Inorder to ensure that the model is only learning localization in the FC layers designated, the weights of the feature layers/CNN layers and the FC layers for classification are froze. This can be done by passing the argument **fix_weights = True** in the model function call in **./models/res_34.py** script in the submission. During the first 15 iterations only localization loss was feed backed to the model and the loss is based on MSEloss(L2 Norm loss function).

Table 3: Model Performance

Model	Fix Weights	Accuracy	Average IoU
Model-1	False	83	0.11
Model-1	True	96	0.09
Model-2	False	94	0.29
Model-2	True	96	0.21

From the results we can seen that the model's performance is better in the model architecture 2. So the Accuracy, IoU and training profiles of models is shown in Figure 7. The accuracy of model trained with fixed weights ensures that the performance remains consistent over all epochs. However, it offers less flexibility and hence the model is not capable of learning the bounding box and the final IoU(Intersection of Union) hits at 21%. And when the model is set to train the whole network, we can see the drop in accuracy and better IoU of 29%, yet model is not able to achieve significant increase in performance in localization.

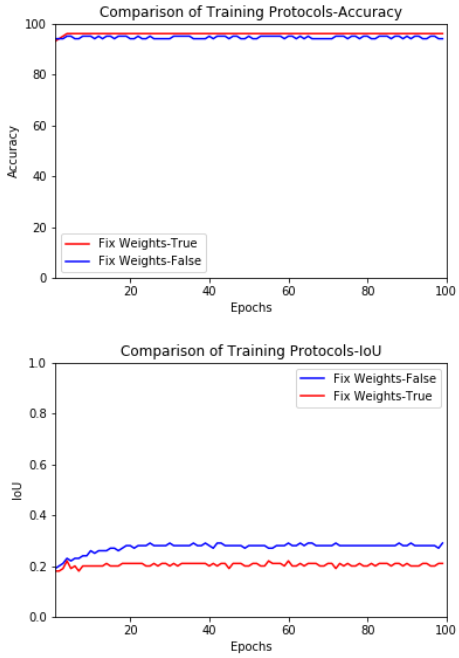


Figure 7: Model-2 Training Accuracy and IoU over 100 epochs.

The loss profile for model-2's training loss is displayed in Figure 8. From the Classification loss, 3.5 is subtracted for visualization purpose and consistent training purpose which is also reflected in the Total Loss. The dashed line represents the training done using **Fix_Weights = True** and solid line is **Fix_Weights = False** respectively. We can see that the consistency loss is very small and the effect of consistency is not dominant during the training process. The reason is that Classification loss is in the range of [3.65,3.75], Localization loss in the range [0.1, 0.01] and Consistency loss is in the range [0.0015, 0.0008] respectively. Even if the Consistency loss is reducing slowly, the effect of Consistency in the model is training is significantly masked by Classification loss which has higher magnitude.

5. Discussion

As we can see in the results, the model was able to learn the features that helps to determine the classification results perfectly using the 3D Resnet-34 model. The model was able to achieve accuracy of 96% on the training dataset and 80% on the testing dataset. However, the extracted features are not capable of learning the localization co-ordinates effectively. Primary inference is that the last layer of the model with 512 feature mask is a spatio-temporal feature that has complicated responses and activation which is

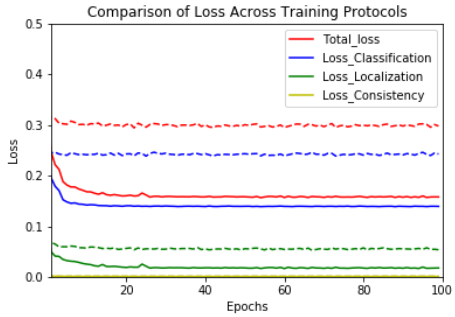


Figure 8: Model-2 Loss

incapable of extracting the bounding box information for the central frame. Also, the dataset is really challenging, for example in the category CliffDiving, TrampolineJumping etc.. the motion of the action bounding box is quick and not confined to a certain region in the video, which makes it really hard to generalize it for the center frame alone by looking at the feature extracted from all the clips. On the other hand, actions like Biking, Basketball etc.. has almost consistent bounding box throughout the clip and action is relaxed, which can generalize the bounding box of the entire clip to its center frame's bounding box response. Thus localization results are sub par and the effects are reflected in the consistency loss and the performance as well.

Inorder to improve the performance of the network, it is vital to be able to extract the features from the center frame of the model that has features corresponding to the activity that is happening in the frame. Available 3D CNN architectures has entangled feature extraction in multiple dimensions which helps determines the action, however not so with the localization task as exact location of action is not as important as the temporal action extraction for the task. In images, the extracted features are confined/localized compared to a clip of images where the type action defines the uncertainty in the localization. Training the model with frozen weights as described in Section 4.3 is proved to be less effective compared to the training of whole network. Also, from the study it is clear that the training of the model with large FC network helped in better localization. An alternative approach is to ensemble the classification model with SSD to detect on the center frame and use consistency loss for both.

The Consistency loss is significantly small compared the localization and classification loss by a factor of 20 with localization loss and 1000 with classification loss. The localization is boosted by a factor and consistency loss is ramped up to make sure that the training is stable and consistent.

6. Conclusion

From the experiments conducted it can be concluded that the model was not able to extract features that distinguishes the boundaries of the action and localize the action for the middle frames. We can see this effect by freezing and unfreezing the feature layers during training. When unfrozen feature layer based model is used, the model tries to learn the localization better but losses classification accuracy when compared to the frozen feature layer model. However, even after significant training, we can see that the loss saturates which indicates that further learning is not possible. The drop in Consistency loss shows that the training setup is suitable for semi-supervised learning, however, the effect of consistency is not significant during the training. The results are not satisfactory when looking at the results of unsupervised clips in the dataset. Model engineering and feature extraction relevant to only the center frame is important to ensure results that are promising.

7. Reference

[1] Jisoo Jeong, Seungeui Lee, Jeesoo Kim Nojun Kwak; "Consistency-based Semi-supervised Learning for Object Detection" Neural Information Processing Systems (NeurIPS), 2019
[2] Kensho Hara, Hirokatsu Kataoka, Yutaka Satoh; "Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?", Computer Vision and Pattern Recognition (CVPR), 2018
[3] Kensho Hara, Hirokatsu Kataoka, Yutaka Satoh.; "Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition". International Conference on Computer Vision (ICCV), 2017
[4] GitHub: https://github.com/kenshohara/3D-ResNets-PyTorch
[5] GitHub: https://github.com/soo89/CSD-SSD

Appendix

All relevant codes used for this experiment to reproduce the result is attached as zip file.