HACETTEPE UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

UNDERGRADUATE PROJECT FINAL REPORT

Project Name	Report Date
Converting Recipe Videos to Text : Action Recognition	17.01.2020

Student Number(s)	Student Name(s)
21527305 21527027	Sevda SAYAN Furkan Çağlar GÜLMEZ
Supervisor(s)	**Company Representative(s)**
Prof.Dr. PINAR DUYGULU ŞAHİN	

Project Coordinator	Report Approval
Date: _____	☐ Yes ☐ No If no, rational of rejection: _____

Project Video Youtube Link
https://www.youtube.com/watch?v=aJVoNHl2-Kk&ab_channel=SevdaSayan

A. TECHNICAL RESULTS

ABSTRACT

In this paper, we introduce Converting Recipe to Text System that recognizes action and detects object in kitchen. Purpose of this work is to find verbs, nouns and finally predict actions for each video segment. We use Epic Kitchens dataset that contains 32 different kitchens, 11,5M frames, 39594 actions segments, 454255 object bounding boxes, 125 verb classes, 331 noun classes. To solve this problem we took an advantage of object detection features and 3D Convolutional Neural Networks (CNN). These methods enhanced accuracy of noun and verb prediction. Our target is to recognize verb, noun pairs to detect action using Temporal Segment Network, Temporal Relational Network, Multiscale Temporal Relational Network. First plan is to pick the model which is the most successful to use it in the rest of the progress. After that extracting feature with I3D, compare them with picked model, find accuracy weights according to actions and merge all of them with object detection results.

Keywords: Video Captioning, Computer Vision, Transfer Learning, Epic Kitchen, Action Recognition, 3D CNN, I3D

I. INTRODUCTION

Analysis of videos by using first-people videos is very topical challenge nowadays. By taking out these approaches, we are aiming to perform a video analyzer for kitchen activities cutting, baking, serving etc. Primary goal is extracting noun and verb couples by using pre-trained models published by Epic-Kitchens team. Our dataset is the largest dataset so far. It contains 11.5 million frames that is about kitchen activities. There are three models based on ResNet50, Temporal Segment Network, Temporal Relational Network and Multiscale Temporal Relational Network. Our first goal was determining which model has highest perpormance. After determine a model that has the most accurate results, we extracted action features by using I3D network published by Brian Zhang, Joao Carreira, Viorica Patraucean, Diego de Las Casas, Chloe Hillier, and Andrew Zisserman. They released Inception-v1 I3D models trained on the Kinetics dataset training split. We finetuned and retrained this model on our dataset with extracted action features by using rgb samples. Purpose of training a network from scratch was improving pretrained model's performances. To do this, these two model evaluated and output weights are associated to obtain better results.

II. BACKGROUND

We have developed our project mainly using Python and PyTorch. We used these two because production is much more faster than other options and considering our knowledge we decided that we are faster with Python. Additionally, PyTorch is a machine learning framework that accelerates the processes of development and its community is big enough by this way we would fix our torch related errors easily so that it is appropriate for this project.

On the other hand, we use such libraries as numpy, pandas to hold our vectors and matplotlib to visualize our results as graphs, PIL library for image loading and processing, time library for calculating train, validation, test processes' time, I3D to train our model to improve pretrained model accuracy.

Actually, original version of I3D implemented using tensorflow[7] but we used torch version[8] of it because we were already using pytorch and we don't have time to get used to tensorflow.

Advantages of PyTorch[9]

Pytorch is easy to learn and easy to code. For the lovers of oop programming, torch.nn.Module allows for creating reusable code which is very developer friendly. Pytorch is great for rapid prototyping especially for small-scale or academic projects.
It's a Python-based scientific computing package targeted at two sets of audiences:

- A replacement for NumPy to use the power of GPUs
- A deep learning research platform that provides maximum flexibility and speed

Tensors : Similar to NumPy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing. It has some operations for matrixes like rand, zeros, add eg. similar to NumPy. Also we can construct a tensor directly from NumPy if we need and vise versa.

CUDA Tensors : Tensors can be moved onto any device using the .to method. And it helps us to do our studies on gpu.

III. RELATED WORK

This type of classification problem is common, because of the raising data resources like on YouTube. We found many documentation about video-classification in the internet. Authors in different documentation used 3D CNN, 2-stream CNN, RNN, VLAD, HOG, HOF. The focus of this study is on recent studies in the field of action / activity detection on food videos (Kuehne et al., 2015; Li et al., 2015; Soran et al., 2015). Kuehne et al., (2015) Activity determination was made from images taken with fixed cameras using Fisher Vectors. Li et al. (2015) Images taken from Egocentric cameras have been removed from motion and object attributes. Soran (2015) aimed to generate alerts for skipped or incomplete steps on recipe videos recorded with person-centered cameras. Huang (2017) have worked on linking the so-called source resolution (resolution reference resolution) to content-related items based on noise, incompatible or shifted subtitle problems generated by DIY-type instructional videos. Zhou et al. (2018) prepared a new dataset consisting of recipe videos collected from YouCook2 web site and identified the independent stages in a recipe video on this dataset and identified the problem of attempting to segment it and tried to solve it.

There is a lot of work related action recognition. One of these work was in recent years [10]. In their work, they introduced that 3D CNN that includes two-stream structure. Therefore, With spatiotemporal regions, they aimed to improve their model using EGTEA GAZE+ dataset[11].

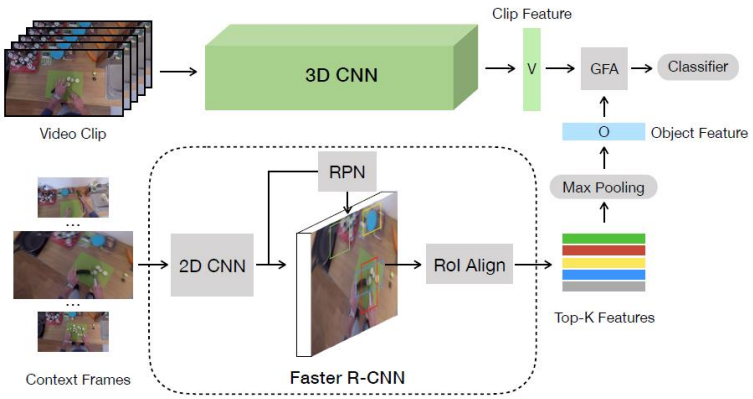
While they are using these, they recognized[3][4] that 3D CNN is rather than using 2D.Also while creating segmentation structure to obtain optical flow they[5] used 2-stream CNN.

On the other hand, authors in [6] used RNN while working with long videos and they found that this is substantial and appropriate for this classification type.

Our state of the art is documentation[1].

This article makes both object detection and action recognition. Since we are focusing on action recognition in our project, we will explain what has been done in that part.

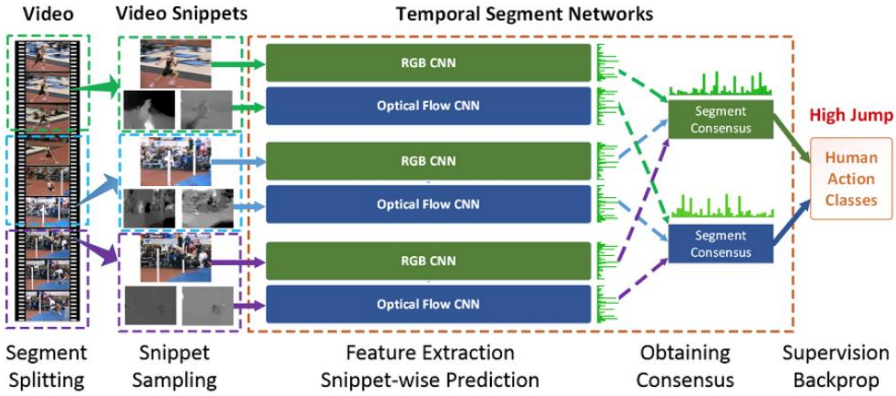
The EPIC-Kitchens data set contains various small objects, intense motion blur and blockages. It is used to guide the training of 3D Convolution Neural Networks (CNNs), which can significantly improve the accuracy of its prediction. The 3D CNN portion takes the video clip as input and creates a clip property. These clip properties are sent to the GFA module. The output of GFA is our last feature and is used to classify verbs. The article we have discussed is stated in the reference section. [1]



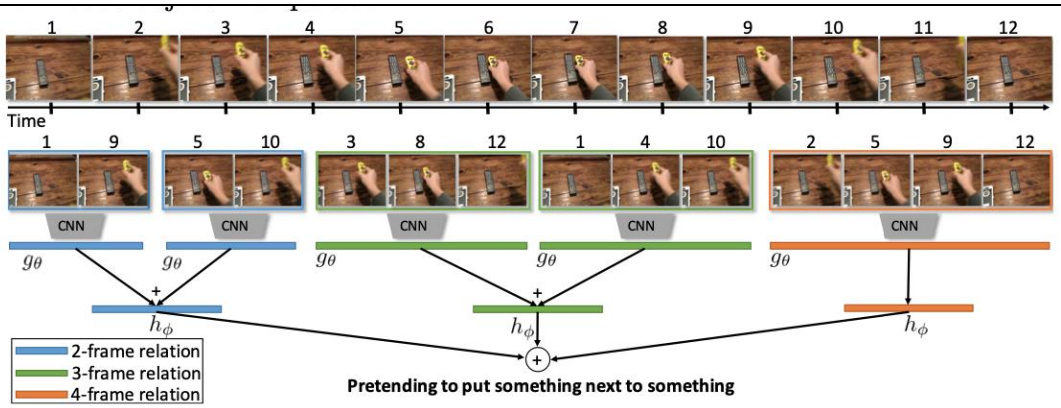
State-of-the-art System [1]

- 3D CNN takes transformed and loaded video clip as input and produces a clip feature.
- Faster R-CNN branch uses 2D CNN and Rol Align and takes context frames as inputs to produce object related features using Top-K features approach.
- GFA takes extracted clip features and object features as inputs and provide an output to recognize action and detect the object. Instead of add two weight, using GFA that they introduce first, has improved their performance.

On the other hand, authors[2] from Epic Kitchens introduced Temporal Segment Network(TSN),Temporal Relation Network(TRN-MTRN) which trained with all train data in Epic Kitchens dataset.

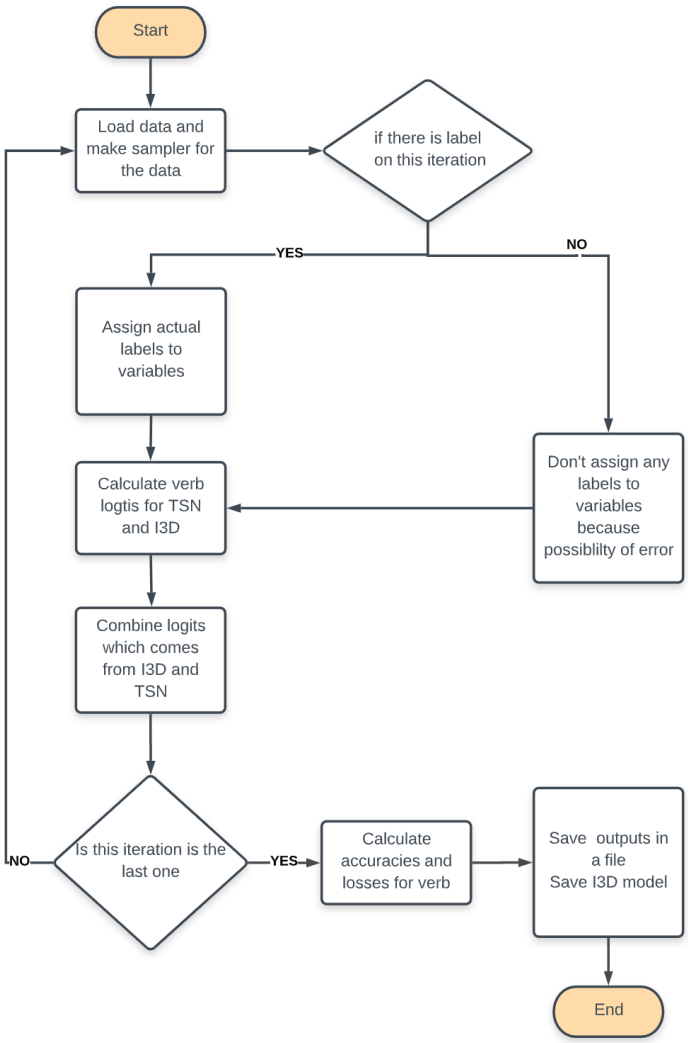


This is working principle of their TSN that is framework to recognize the action based on videos. It learn with segment pairs of videos and each segment includes 8 frames. This is also called long-range temporal dynamics and uses RGB diff & warp flow as new techniques.



This is working principle of their TRN & MTRN that use reinforcement learning and multi-layer perceptrons. Basically model is pretending to put something next to something and takes a specific number of ordered frames from video as inputs. Lastly, its output is likely a type of CNN outputs from activation functions.

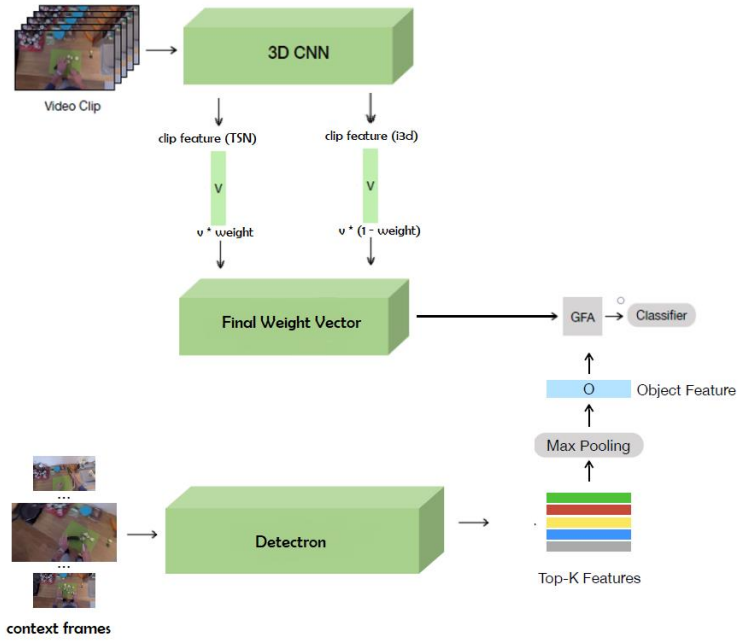
IV. METHOD



Our background is like above.

- Firstly, we load data frames and using these data frames, create data sampler to use it while extracting action features.
- Then we iterate this data sampler one by one querying each of them, if any label returns from queries, assign our local variables to returned labels.
- Following, we combine calculated TSN and I3D logits by multiplying them with two scalar one of them is scalar, other one is 1-scalar.
- Afterwards this process works until last sampler, when last sampler comes and finished its process, losses and accuracies is calculated.
- Lastly, video informations and outputs saved in a file and model is saved with .pt extension.

V. TECHNICAL DESIGN AND CONFIGURATION



Step – 01: Take 8 segment of images out of 200 images from video clip for TSN.

Images as inputs should be in format torch tensor for TSN.

Torch tensor for an image should be in a shape: [1, 3, 224, 224]

Value 3 in shape stands for Red, Green, Blue.

Value 224, 224 stands for WidthxHeight of image.

Torch tensor for 8 segmented image should be in a shape: [1, 24, 224, 224]

Value 24: There should be 8 image for segmentation count 8 so 3 multiplied by 8 resulted 24.

Step – 02: Take 79 segment of images out of 200 images from video clip for I3D.

Images as inputs should be in format torch tensor for I3D.

Torch tensor for an image should be in a shape: [1, 3, 224, 224]

Value 3 in shape stands for Red, Green, Blue.

Value 224, 224 stands for WidthxHeight of image.

Torch tensor for 79 segmented image should be in a shape: [1, 3, 79, 224, 224]

Value 79 in shape stands for segment count to build optical flow.

Step – 03: Give each segments 3D CNN (TSN & I3D) as an input.

TSN input: [1, 24, 224, 224]

I3D input: [1, 3, 79, 224, 224]

Input image should have been normalized to size 224x224 before feeding models.

Step – 04: Calculate weights_tsn * s and weights_i3d * (1 - s) for each index of related vectors. And calculate their sum.

weights_tsn and weights_i3d should be in a shape: [1, 125] and its type is Torch Tensor.

125 stands for count of verb classes.

And both tsn and i3d weights must be [1, 125] because after calculation we take sum of them to create new weighted vector.

Step – 05: Take maximum index for Top1.

Step – 06: Take maximum 5 index for Top5.

Step – 07: Evaluate unseen data.

Unseen dataset must have annotations in order to calculate losses and accuracies so that never test with Epic Kitchens test set, because test set do not have annotations. Otherwise, you should test combined model using unseen labeled dataset.

Step – 08: Take context frames from video clip for Detectron.

Step – 09: Give context frames to Detectron as an input.

Step – 10: Extract Top-K Features using Detectron.

Step – 11: Apply max-pooling on Top-K Features and result is Object Feature.

Step – 12: Merge verb results with object results.

Step – 13: Finally, classify an action.

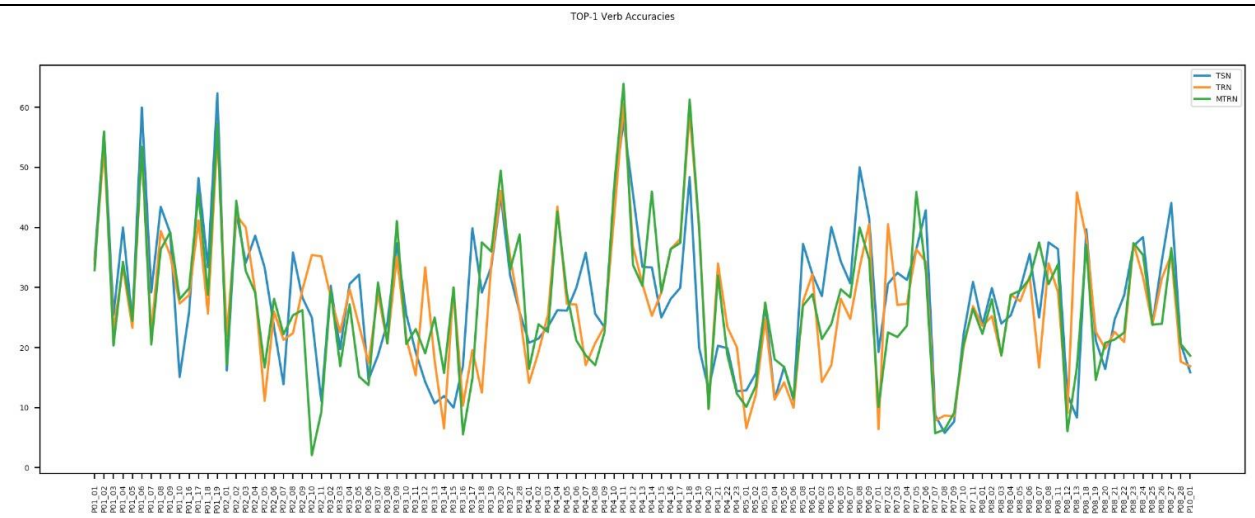
Returned value should be in a format integer,integer. For example 1,54 while 1 means verb put and 54 means noun salad so that action would be "Put Salad"

VI. PROJECT IMPLEMENTATION

Seen Kitchens (S1)

#	User	Entries	Date of Last Entry	Team Name	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Precision (%)			Recall (%)		
					Verb ▲	Noun ▲	Action ▲	Verb ▲	Noun ▲	Action ▲	Verb ▲	Noun ▲	Action ▲	Verb ▲	Noun ▲	Action ▲
4	EPIC_TSN_RGB	1	09/05/18		31.81 (3)	16.22 (5)	6.00 (4)	76.56 (3)	42.15 (4)	18.21 (4)	23.91 (2)	19.13 (4)	3.47 (4)	9.33 (4)	11.93 (5)	2.64 (5)
7	EPIC_TSN_Fusion	1	09/05/18		30.66 (7)	14.86 (6)	4.62 (7)	75.32 (6)	40.11 (6)	16.01 (6)	8.84 (8)	21.85 (2)	2.49 (6)	6.76 (8)	9.15 (7)	1.72 (7)
8	EPIC_TSN_Flow	1	09/05/18		29.64 (8)	10.30 (8)	2.93 (8)	73.70 (7)	30.09 (8)	10.92 (8)	18.34 (5)	10.70 (7)	1.56 (7)	6.99 (7)	5.48 (8)	1.11 (8)

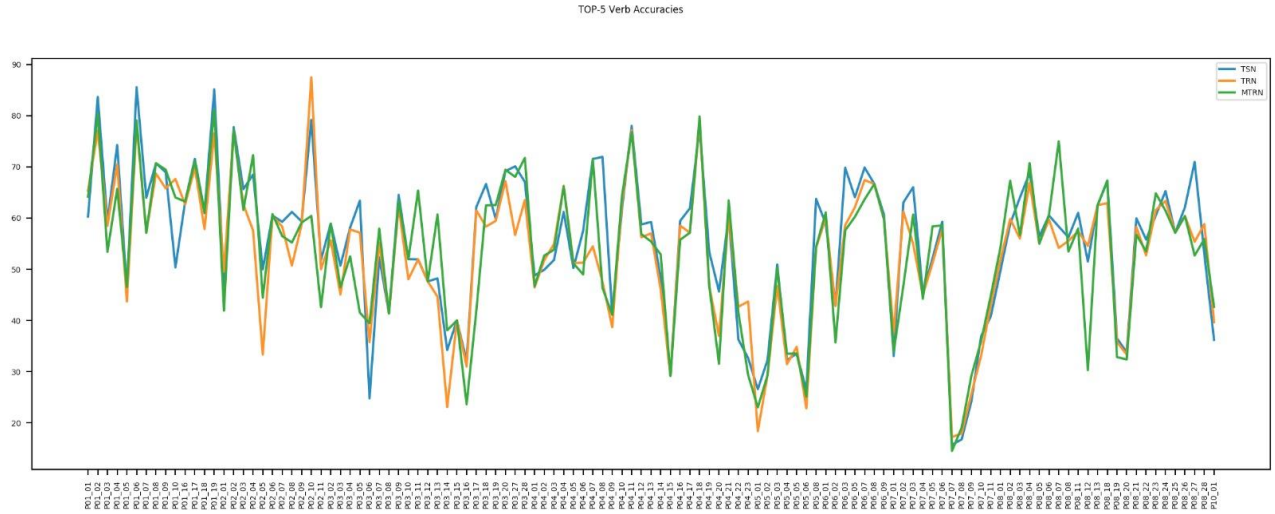
First of all, in the development process of our project, to decide what the baseline is, we decided to use pretrained models of Epic Kitchens. We tested seen data with TSN, TRN, M-TRN one-by-one to select one of them which we will improve. So, our first goal was determining which model has highest perpormance. Figure that stated above shows results of seen kitchens with pretrained TSN of Epic Kitchen's authors[2]. To improve these results and getting detailed total and class based accuracies, this three 3DCNN models are evaluated.



This figure shows top1 accuracy of each TSN, TRN, M-TRN.
It ranges from min 10% to max 60%. Therefore, we calculated their average accuracies.

Average top1 accuracy of TSN : 28.35529%
Average top1 accuracy of TRN : 27.15994%
Average top1 accuracy of mTRN: 27.08505%

Because of that they were so close to each other, we thought that we put one observing parameter. Then we realized that all of them decrease and increase at same video but TSN decrease less and increase more and also TSN's average accuracy is 1.2% more than others so that we decided to use a pretrained TSN model after we combined this model with I3D.



This figure shows that each TSN, TRN, M-TRN top5 accuracies.

Average top5 accuracy of TSN : 55.24341642678649%
Average top5 accuracy of TRN : 53.1399713047717%
Average top5 accuracy of mTRN: 53.549403893009575%

Model	Metrik	P01	P02	P03	P04	P05	P06	P07	P09	P10	Ort.
TSN	İlk 1	**38.33**	27.48	24.85	28.96	18.89	**36.78**	24.37	**28.03**	15.89	**28.35**
	İlk 5	**67.44**	61.80	53.10	**55.05**	37.94	**61.82**	41.20	57.73	36.18	**55.24**
TRN	İlk 1	35.38	**28.57**	24.50	30.56	15.22	27.18	22.18	26.93	16.88	27.15
	İlk 5	65.68	58.69	50.63	52.88	33.96	59.63	40.43	56.04	**59.69**	53.13
M-TRN	İlk 1	36.22	23.04	25.89	**30.95**	17.79	29.57	20.45	26.24	**18.64**	27.08
	İlk 5	66.22	57.46	52.62	53.00	35.57	57.81	40.56	56.32	42.65	53.54
I3D	İlk 1	24.9	16.2	**34.9**	16.9	**19.2**	21.6	**34.7**	23.8	17.3	20.9
	İlk 5	59.6	**69.7**	**65.7**	49.5	**63.9**	49.2	**67.3**	**64.3**	43.5	52.0

Average Accuracy for 3D Models

As it seen in the figure which top1 results and top5 results and we talked about top1 results, top5 accuracies are same as top1 results but its range is from 30% to 80% but their average is 55-53%. TSN average top5 accuracy have 2.1% more than others have.

- **INCLUDING MANY SHOT ACTIONS**

In order to improve action results, we thought that object detection is making a lot contribution. However, our actual aim is action recognition so that because of that our time is limited, we did not have time to waste it with object detection. In order to make an assumption how it is affects the action recognition, we took ground truth labels into account.

When we are testing the data, we obtain noun classes with noun labels file. Afterwards, to observe how it is affected the results, we should take one role into playing which named many shot actions that is described as most used verb noun pair in the dataset. By querying many shot actions using noun class ids, we also obtain most used verb noun pairs and we tried to get correct results.

But in the end of this experiment, we recognized that it harms to our model and decreases its accuracy. Consequently, we decided not to use this method.

- **INCLUDING I3D MODEL**

After all of this, we observe that the pretrained TSN is the one overwhelm others so that we decided to improve TSN. Unfortunately, unseen data of Epic Kitchens does not have annotations. Therefore, we trained Temporal Segment Network from scratch and to improve results of our model, we also trained an I3D model using PyTorch. Because our aim to train I3D model was to combine it with TSN.

While training I3D, beginning goal was to find time interval to look for an action. Two kind of time interval is used. First one is about 3s or in other words 200 frames and the other one is abaut 1.5s or in other words 100 frames while videos are type of 60 frame per second.

	Epoch #1	Epoch #2	Epoch #3	Epoch #4	Epoch #5	Epoch #6	Epoch #7
Train Accuracy on segment 79/200	0.1715	0.3090	0.3020	0.2873	0.3274	0.3291	0.3432
Train Loss on segment 79/200	2.759	2.5255	2.4756	2.3280	2.1392	2.1008	1.9012
Validation Accuracy on segment 79/200	0.980	0.3831	0.0260	0.4387	0.2968	0.4967	0.3243
Validation Loss on segment 79/200	46.7504	42.4763	3.9199	3.2735	12.8868	7.8525	25.5042

	Epoch #1	Epoch #2	Epoch #3	Epoch #4	Epoch #5	Epoch #6	Epoch #7
Train Accuracy on segment 32/100	0.1134	0.3766	0.3885	0.3907	0.3109	0.3898	0.2810
Train Loss on segment 32/100	2.6470	2.3973	2.2951	2.3055	2.2437	2.2374	2.6067
Validation Accuracy on segment 32/100	0.0952	0.1089	0.1357	0.1289	0.1662	0.2052	0.1599
Validation Loss on segment 32/100	144.6351	104.5937	34.6025	27.4033	13.9647	10.2521	5.2911

As seen on these results above, best validation accuracy on 200 frame is 0.4967 while best validation accuracy is 0.2052 on 100 frame. Based on this observation we decided to keep going with 79 segment in 200 frames as an input.

According to above observations, we trained I3D on all dataset (nearly 4M frames). In this process we used Adam optimizer and learning rate is setted to 0.001. Each experiments runs for a total of 10 epochs, where epoch refers to one cycle through the full training dataset. Training process is was in progress cumulatively or in other words it handled kitchen by kitchen because of the machine time so first P01 is trained and then P02 and so on. When we think as all of the process one piece each epoch took time about 23 hour and it almost took 10 days for all training and evaluating process. On each epoch we usually observed decreasing losses and increasing accuracies as it should be.

	P01	P02	P03	P04	P05	P06	P07	P08	P10	AVG
TOP1	24.9	16.2	34.9	16.9	19.2	21.6	35.7	23.8	17.3	20.59
TOP5	59.6	69.7	65.7	49.4	63.9	49.2	67.3	64.3	43.5	52.00

Combining weights of TSN and I3D results was substantial for improvement. Output weights of I3D multiplied with a alpha and likewise output weights of TSN multiplied with (1-alpha).

$$Model = \alpha * I3D + (1 - \alpha) * TSN$$

Finally all weights are associated like **GFA** described in **state-of-the-art work**. The problem here was choosing right alpha to integrate these two networks. To find fittest alpha some experiments performed on final models.

Alpha = 0.2

	TOP1	**TOP5**
I3D	16.41%	50.51%
TSN	26.25%	71.25%
Final Weighted Model	28.74%	66.83%

When scalar value is set to 0.2, our baseline which is top1 accuracy of TSN is improved by 2.5%
But this value could not improve top5 accuracy of TSN unlikely it decreased by 4.5%.
So 0.2 is not a better value to select.

Alpha = 0.65

	TOP1	**TOP5**
I3D	17.26%	50.59%
TSN	26.29%	73.42%
Final Weighted Model	19.25%	60.47%

Alpha 0.65 was the worst value we ever tried. There was improvement neither top1 nor top5. Therefore deciding the value is this was unreasonable.

Alpha = 0.4

	TOP1	**TOP5**
I3D	14.51%	51.02%
TSN	23.54%	72.69%
Final Weighted Model	24.06%	64.33%

Alpha value 0.4 is likely value 0.2. It increased top1 accuracy but top5. This is unreasonable to select until we find a better value.

Alpha = 0.1

	TOP1	**TOP5**
I3D	18.13%	54.84%
TSN	27.02%	74.09%
Final Weighted Model	34.45%	76.91%

When we look at these results, when use 0.2, 0.4, 0.1 values top1 accuracies are increased in the order %2, %1, %7 and top5 accuracies are decreased with scalars 0.2 and 0.4 but 0.1 is increased 2%.

Considering all of these observations **the best value is 0.1**. It **improved** both top1 and top5 accuracy of TSN more than other alpha values.

VALIDATION AND RESULTS

```
#--------------EVALUATING I3D----------------------
print("I3D Input Shape : ", input_i3d.shape)
rbg_score, rgb_logits = model_i3d(input_i3d)
print("I3D Output Shape : " ,rgb_logits.shape)
res_verb_i3d, ind_verb_i3d = rgb_logits.topk(5)
for indice in ind_verb_i3d[0]:
   if indice.item() == actual_verb_class:
        success_verb_i3d_top5 += 1

verb_value_i3d, verb_indice_i3d = torch.max(rgb_logits, 1)
predicted_verb_class_id_i3d = verb_indice_i3d[0].item()


#--------------EVALUATING TSN----------------------
model_tsn.eval()
model_tsn.requires_grad_(requires_grad=False)
print("\nTSN Input Shape : ", input_tsn.shape)
output_tsn = model_tsn.features(input_tsn)
verb_logits_tsn, noun_logits_tsn = model_tsn.logits(output_tsn)
print("TSN Output Shape : ", verb_logits_tsn.shape)

scalar = 0.1
final_verb_logits = torch.add( torch.mul(rgb_logits, scalar) , torch.mul(verb_logits_tsn , 1-scalar) )
print("\nFinal Verb Logit Shape : " ,final_verb_logits.shape)
```

Our code for evaluating I3D and TSN is stated above.

- While evaluating **I3D**, to validation of input shapes we printed **input_i3d.shape** which is **[1, 3, 79, 224, 224]** and we give this input to our I3D model named **model_i3d** that returns two output as **rgb_score** and **rgb_logits**.
- rgb_logits stands for **extracted action features** output whose shape is **[1,125]** and type is torch tensor.
- After that, using torch's **topk** functions with **rgb_logits** we obtained **top5** indices and torch's max function with rgb_logits we obtained top1 indices.
- **Evaluation** process of TSN almost the same as I3D but differ from 2 way. First one is that **TSN returns noun_logits** too and other one is that TSN's input shape is **[1, 1, 24, 224, 224]** because of **8 segmentation**.
- TSN's and I3D's output shape returned from evaluation have same shape, therefore we can add them after multiplied them with scalar and 1-alpha.
- Lastly, **final_verb_logits'** shape **is [1, 125]** stands for 125 predicted verb indices.

```
I3D Input Shape :  torch.Size([1, 3, 79, 224, 224])
I3D Output Shape :  torch.Size([1, 125])

TSN Input Shape :  torch.Size([1, 1, 24, 224, 224])
TSN Output Shape :  torch.Size([1, 125])

Final Verb Logit Shape :  torch.Size([1, 125])
```

Steps that is stated above is validated by this output.

```
verb_value, verb_indice = torch.max(final_verb_logits, 1)
predicted_verb_class_id = verb_indice[0].item()

noun_value_tsn, noun_indice_tsn = torch.max(noun_logits_tsn, 1)
predicted_noun_class_id_tsn = noun_indice_tsn[0].item()

print("Predicted result is (", predicted_verb_class_id, ' , ',predicted_noun_class_id_tsn, ")")
print("Result stands for ", verb_dict[predicted_verb_class_id] , ' ' , noun_dict[predicted_noun_class_id_tsn)
```

- As an end to evaluation process, last step is predict an action using top1 outputs above.
- Verb_value calculated using weights which combined using I3D and TSN returns value 1 for verb class and noun_value_tsn returns value 54 for noun class.
- Value 1 is found from verb annotation file and value 54 is found from noun annotation file whose definitions are put and salad.

```
Predicted result is (1,54)
Result stands for put salad
```

Steps that are stated above is validated by this output.

For this frame stated on left our model predicts "Wash Carrot" output.

For this frame stated on above our model predicts "Dice Courgette" output.

As a result, the accuracy rate of the predicted actions on pretrained models was increased but could not be brought to the desired level. The main reason for this was the excess of the kinds of actions in the data set and the similarity of these actions. For example, the actions of ’open’ and ’close’ were very similar in flow, and also these types of actions was usually on the same objects. Due to the excess of these and similar actions, the model did not gain the ability to predict such actions correctly. In addition to this problem, since the videos contain every movement in the kitchen at that moment, there are too many sections without any action. Since such cross-sections cannot be distinguished, the performance of the model was exceedingly reduced.

Figure: Action of close

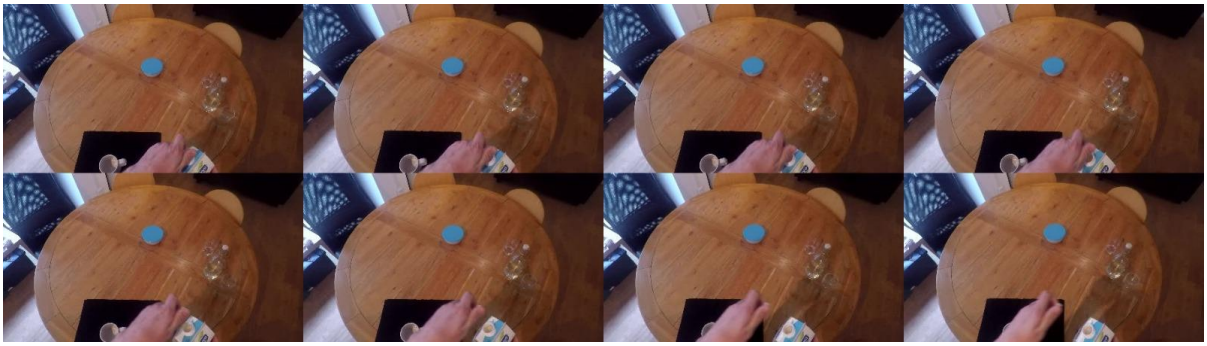


Figure: Action of open

CONTRIBUTION(S) TO INDUSTRY AND ECONOMY

As the contribution of our project to the sector;
-Possibility of searching the video from the internet by making use of the movements in the video,
-People who not have enough opportunity time to watching videos, can access the text of video,
-A user with limited internet, can access the recipe by looking at the text rather than watching the video.
-Extracted recipes from this projects may used for feeding Natural Language Processing models in Companies whose interesting is food or recipes.

INNOVATIVE ASPECTS

As an innovation, our work can be integrated into global sites such as youtube to take a step towards making video searches and finding videos easier. User watched video from the internet and if user wants to find it again, he will have the opportunity to find the video again by searching for the movements in the video.

REFERENCES

[1] Baidu-UTS Submission to the EPIC-Kitchens Challenge 2019

[2] https://epic-kitchens.github.io/2019

[3] Lubomir Bourdev, Lorenzo Torresani, Rob Fergus, Du Tran, Manohar Paluri. Learning spatiotemporal features with 3D CNN in ICCV, 2015.

[4] Andrew Zisserman, Joa Carreira. Kinetics dataset and new model in CVPR 2017.

[5] Andrew Zisserman, Karen Simonyan. 2-Stream CNN for action recognition, in Neurips 2014.

[6] Ross Girshick, Piotr Dollar, Zhuowen Tu, Saining Xie, Kaiming He. Residual transformations for neural networks in CVPR, 2017.

[7] https://github.com/deepmind/kinetics-i3d

[8] https://github.com/rimchang/kinetics-i3d-Pytorch

[9] https://medium.com/bbm406f19/week-3-plant-disease-detection-6518ad8c7ef2

[10] Minlong Lu, Danping Liao, Ze-Nian Li, Learning Spatiotemporal Attention for Egocentric Action Recognition

[11] http://ai.stanford.edu/~alireza/GTEA_Gaze_Website/GTEA_Gaze.html

B. PROJECT RESULTS

I. CHANGES TO PROJECT PLAN

Only change was I3D and point was extracting action features using it. By this way, we improved our result by combining I3D onto TSN.

II. PROJECT MILESTONES AND OBJECTIVES

Milestone #	Primary Objective	Due Date	Project Deliverable (if any)	Milestone Achieved?
1	Scanning literature and making a decision for which of them will be used	31.08.2019	-	YES
2	Make a decision of which dataset will be used	07.09.2019	-	YES
3	Downloading pretrained models, preparing data and observe top1 and top5 accuracies	09.11.2019	First Results	YES
4	DEL#2: Project Progress Report	13.12.2019		YES
5	I3D research for improving our TSN model	15.12.2019		YES
6	Make decision if we should use I3D	16.12.2019		YES
7	Selecting I3D and integrating it into our project	20.12.2019		YES
8	Train I3D to extract action features	25.12.2019		YES
9	Test trained I3D model to validate if it worth to train process	30.12.2019		YES
5	Improve model by merging our action results with object detection results and getting a better accuracies	05.01.2020	Project Final Report & Final Demo	YES
6	DEL#3: Project Final Report	17.01.2020		YES
7	Preparing a Turkish paper for SIU	17.01.2020		YES

III. PROJECT PRACTICES AND MEASURES

Task #	Task Description	Responsibility	Start Date	Finish Date	Success Criteria	Task Succeeded ?
1	Researching about related works	Sevda SAYAN	22.08.2019	31.08.2019	Finding most related about our topic and most accurate works	YES
2	Inzva#1st / Deciding Meetup Topics	Furkan & Sevda	24.08.2019	31.08.2019	Meetup topics must be decided	YES
3	Inzva#2nd / Literature Review Results, Deciding Baseline methods	Furkan & Sevda	24.08.2019	31.08.2019	Literature review must be done and reviewing baseline methods process started	YES
4	Inzva#3rd / Presenting baseline results, New approach proposals.	Furkan & Sevda	31.08.2019	21.09.2019	Baseline results must be presented	YES
5	Researching about datasets can be used for kitchen activities	Furkan Caglar GULMEZ	01.09.2019	07.09.2019	Dataset must have detailed annotations and sufficient number of inputs for training	YES
6	Downloading whole dataset into Hacettepe Computer Vision servers	Furkan Caglar GULMEZ	08.09.2019	11.09.2019	Tar files must be extracted and type of folders must be arranged properly	YES
7	Finding and searching about pretrained models to choose, decide with one we will use	Sevda SAYAN	12.09.2019	21.09.2019	The model that we choose must give most accurate results	No, we decided to choose it after we try all of them.
8	Inzva#4th / New approach proposals presenting	Furkan & Sevda	21.09.2019	12.10.2019	New approaches must be presented	YES

9	Writing our own DataLoader	Sevda SAYAN	22.09.2019	16.10.2019	Input sizes must be appropriate for Resnet50, labels must be extracted from annotation files properly	YES
10	Inzva#5th / Complete project presentation, Showcase trial	Furkan & Sevda	12.10.2019	02.11.2019	Presentations must be completed and make a trial for showcase	YES
11	Writing our own Sampler	Furkan Caglar GULMEZ	17.10.2019	07.11.2019	Each model must take 8 segment of images every time for extracting optical flow	YES
12	Inzva#6th / Project Presentation Day	Furkan & Sevda	02.11.2019	09.11.2019	Project presentations must be presented	YES
13	Extracting top1 accuracies	Sevda SAYAN	08.11.2019	20.11.2019	Accuracies must be extracted truthfully	YES
14	Extracting top5 accuracies	Sevda SAYAN	21.11.2019	04.12.2019	Accuracies must be extracted truthfully	YES
15	Observing whole top1 and top5 results into server for each model by using whole dataset that downloaded to server	Furkan Caglar GULMEZ	05.12.2019	12.12.2019	Results must be saved correctly and properly	YES
16	Researching about I3D	Furkan Caglar Gulmez	12.12.2019	15.12.2019	Appropritate I3D model with pretrained TSN model must be exist	YES
17	Making decision if we should use I3D or not according related works	Furkan Caglar Gulmez	15.12.2019	16.12.2019	I3D model must solve the problem	YES

18	Integrating I3D to our project appropriately	Sevda SAYAN	16.12.2019	20.12.2019	Model must be easy to integrate to our project	YES
19	Preparing Final report for DEL#3	Furkan & Sevda	01.01.2020	17.01.2020	Final paper must be in Design Project Final Report Format	YES
20	Preparing SIU Paper	Furkan & Sevda	16.01.2020	17.01.2020	SIU paper must be in SIU format	YES

Team Member	Task # Under Responsibility	Description of the Work Done
Sevda SAYAN	1, 7, 9, 13, 14, 18, 19, 20, 2, 3, 4, 8, 10, 12	Dealing with related works, researching pretrained models, loading dataset properly, calculating accuracies and general codes. Preparing and working for presentation in Inzva.
Furkan Caglar GULMEZ	5, 6, 11, 15, 16, 17, 19, 20, 2, 3, 4, 8, 10, 12	Dealing with server issues, deciding dataset among several numbers of datasets, writing Sampler class to Dataloader and helper codes that will integrated into project. Test models in server with all data and make an analysis for them. Preparing and working for presentation in Inzva.

IV. PROJECT BUDGET

Item #	Description of Expense	Date of Expense	Planned Amount	Actual Amount	Amount Difference
1	We would buy a server to train complicated model with huge dataset	-	50$	0	-50$

First, we thought that we should buy server to train our model because of our dataset is huge enough to buy a server. But then, our teacher opened an account on Computer Vision Servers.

V. PROJECT RISKS

Risk Item #	Description	Probability	Effect	Did It Happen?	How did you handle its occurrence if happened? (Plan-B)
1	For use the dataset, we can't have enough GPU	High	It affects our project directly	YES	We raised our problems with our advisor and ask her if she give us a server account

2	Some problems that we do not take into account in our project may occur and we may fall behind the schedule	Medium	Our business may have to be postponed and we may miss the deadlines for us.	YES	By doing more work in a short time we accelerate the calendar and tried to keep up with our plans. We gave priority to important tasks.
3	The results of our project are not as good as we expected.	Medium	It lowers our motivation and may lead to low marks	NO	We will try new methods to improve our success and reduce our error rate if it occur
4	We may have problems with our group friend	Low	The progress of the project slows down and the motivation decreases	YES	We try to solve the problem drawlingly.
5	Try improving existing model using I3D model because It depends on machine time.	Medium	I3D was a correct choice.	NO.	Training I3D improved our model.

VI. SELF EVALUATION

We were not well versed in topic of action recognition by using computer vision. Because of that some algorithms that we will use was a wrong decision. These situations cause us to lose a lot of time.

The environment that we should proceed with was a contradiction for us. In the earlier stages we were using Google Colab environment to develop our project by uploading a minimized version of our dataset to Drive. Afterwards we awaken to usefulness of vim on our school's GPU server. After getting used to it our progression started to speed up.

We spent most of our time to deal with rgb data frames to train our model and afterwards flow data frames are decided to use for extracting action features. This made more sense and then we noticed that it was more sensible.

Our dataset was too large and in the beginning of this project we were thinking that this will provide us profits. However because of the dataset`s greatness and profitableness, our final model cannot gain ability to predict all actions properly. We should have choose more efficient dataset.

VII. LESSONS LEARNED

We learnt

- how to use pretrained models,
- how to finetune already trained models,
- how to train a model using 3D CNN from scratch,
- how to combine two trained model to improve performance,
- how optical flow's working principle,
- how to use PyTorch framework,
- how to handle big data,
- how to Python script to automatize works,
- how to use Linux Server especially "screen" command saved our day,
- how to be more planned
- how GPU is substantial for ai project.

We wanted to talked about Inzva community that we joined and present our project.

Also, we learnt different Artificial Intelligence algorithms and methods such as GAN, reinforcement learning.