Action recognition - Task A

This part is about running inferences on real world videos from KFC kitchens. The models used are classifying action (verb + noun) that is performed in the video. First part is to transform data into correct shape to feed the model.

Dataset

After video is downloaded from youtube, all the relevant actions that happened at different timestamps are recorded in the format:

	filename	naration_id	start_time	end_time	verb	verb_class	noun	noun_class
2	KFC_1	0	0:00:05	0:00:10	roll-in	roll	flour	flour
3	KFC_1	1	0:00:16	0:00:18	mix-in	mix	sieve	strainer
4	KFC_1	2	0:00:23	0:00:25	filter-sink	filter	flour	flour

There are 125 verb classes and 352 noun classes, combination of these two gives an action. The test dataset (**D1**) contains 45 labeled video parts, extracted from 5 youtube videos. Columns ***start_time*** and ***end_time*** are important because they show which frames to look at in the video (this depends on FPS (frames per second)). Every video part is split into 8 segments and from each segment one frame (snippet of length 1) is randomly picked. This results in a group of 8 RGB images which are than processed with these operations: **Resize** (keep original ratio between height and width, the smaller edge is 256), **Crop in center** (get central part of the image in shape 224 x 224), **Stack** (convert list of 8 images to shape (*num_channels* * *segment_count* * *snippet_length*) x *height* x *width*), **Convert to torch tensor** (gets torch tensor from PIL data), *Normalize* (subract corresponding mean value from each channel and divide by corresponding std value). These transformation are written in *transforms.py* file and were taken from the official github repository (EPIC KITCHENS).
At last, at position 0 new dimension is added (corresponding to batch size dimension) and each video is fed separately into the model. Input shape of one instance of video is then 1 x 24 x 224 x 224.

Extracting frames from video is done using function `create_inputs_from_annotations` from *dataset.py* file. Videos (KFC_1, 2, 3, 4, 5) are approximately of same fps (30), except KFC_3 which is 25.

Another dataset (**D2**) that is taken is *P01_11.tar* (https://data.bris.ac.uk/data/dataset/b2db579428d236ae3f529ab05d8aa55e/resource/bff828c1-466f-4168-b092-2c0b536013bd?inner_span=True) from validation dataset. This kind of videos are already seen during training (similar to the training set) and it is expected that metrics on them are better than on unseen data (D1). This is used just for the purpose of comparing with our case (100 randomly picked examples are used from validation dataset). Extracting frames from this dataset is done using function `create_inputs_from_annotations_val` from *dataset.py* file.

Here are some of the statistics of videos used:

Dataset	Video name	Avg. action length [sec]	FPS
D1	KFC_1	3.13	30
	KFC_2	2.14	30
	KFC_3	1.67	25
	KFC_4	7.4	30
	KFC_5	4.62	30
D2	P01_11	3.68	60

Table 1: Average length of action in each video

Model comparison

Models compared in this task are TSN, TRN, MTRN and TSM backed with *resnet50* backbone architecture.

D1

Model	Verb Acc @1	Verb Acc @5	Verb F1 weight	Verb Loss	Noun Acc @1	Noun Acc @5	Noun F1 weight	Noun Loss
TSN	11.11%	24.44%	0.044	8.21	2.22%	8.88%	0.022	6.53
TRN	2.22%	**28.88**%	**0.041**	4.38	**2.22%**	**8.88%**	0.011	5.24
MTRN	8.88%	24.44%	0.067	4.50	2.22%	**11.11%**	**0.016**	5.08
TSM	**17.77%**	20%	0.054	8.29	2.22%	8.88%	0.012	6.62

D2

Model	Verb Acc @1	Verb Acc @5	Verb F1 weight	Verb Loss	Noun Acc @1	Noun Acc @5	Noun F1 weight	Noun Loss
TSN	17%	73%	0.10	3.78	0	10%	0.005	6.77
TRN	**30%**	57%	**0.23**	2.93	**4%**	**16%**	**0.018**	5.36
MTRN	21%	60%	0.22	2.96	**6%**	**15%**	**0.019**	5.63
TSM	19%	67%	0.12	3.89	0	10%	0	6.84

Table 2 and 3: Comparison of models on 2 different datasets

Action accuracy is not considered since there are none actions completely classified correctly. TRN and TSM models show best results on verb classification (top 1 accuracy), while TRN and MTRN give higher accuracy on noun recognition task.
Verb Loss and Noun Loss are observed separately as a Cross Entropy Loss on verb/noun logits from the model and it is clear that TRN has the smallest loss from all other models on verb recognition, while MTRN has on noun recognition task.

Looking at D2 results, TRN results in best accuracies for both verb and noun recognition, and MTRN for nouns. As expected, metrics on D2 dataset (seen data) are much better than on D1 (unseen data) since the distribution of data is more similar to the train data. Also, D1 dataset is more specific to some subset of actions (relevant to KFC kitchens) and some fine-tuning to

these kinds of actions might help in improving these accuracies. Moreover, videos from D2 set are of higher FPS so these images are of higher quality and the model can more easily recognize what is happening there.

Nouns and verbs that show up in D1 are presented in Fig. 1.

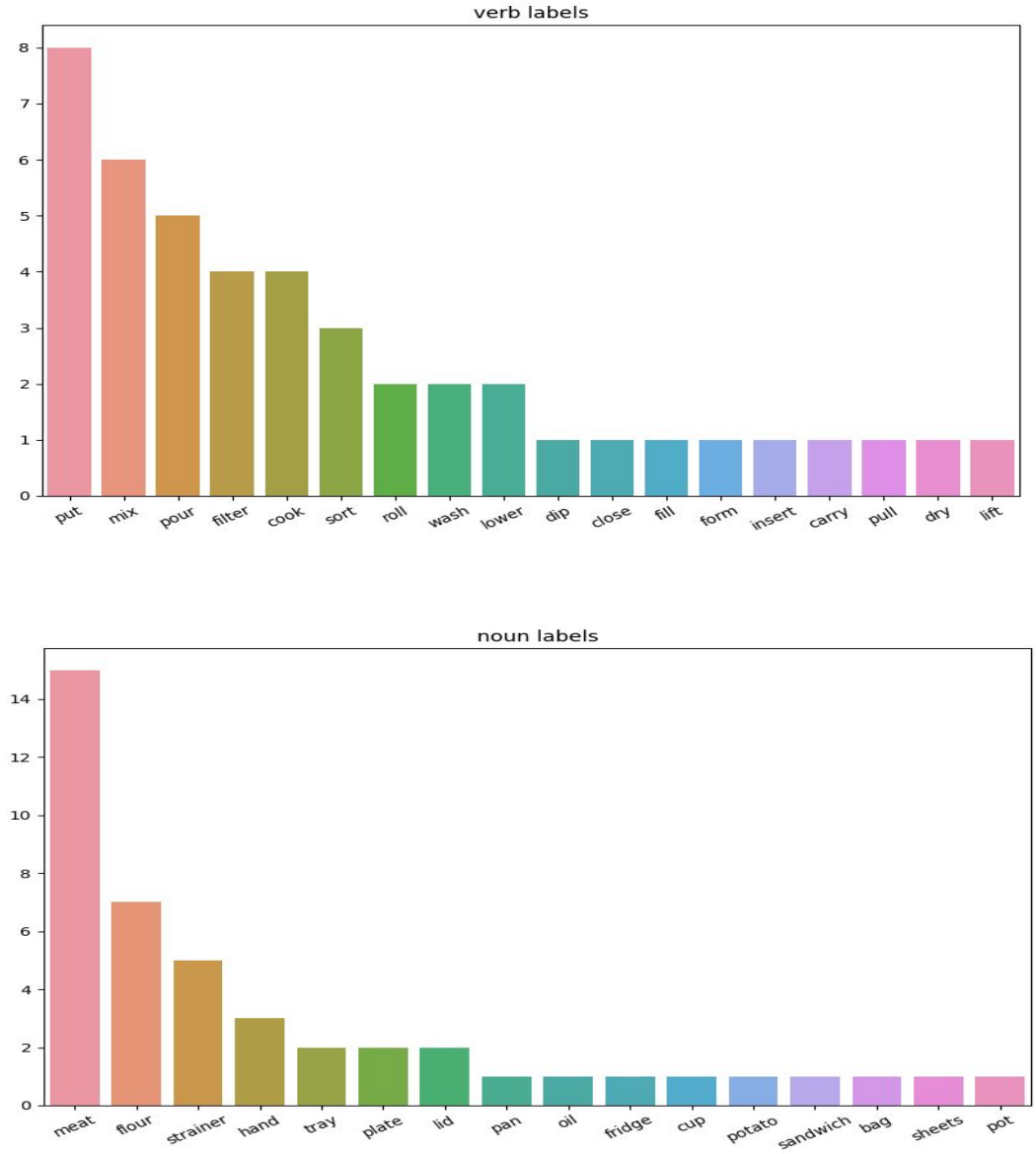

Fig 1: Distribution of most frequent verbs and nouns labels in test dataset D1

It is obvious that some verbs (put, mix, pour) and nouns (meat, flour, strainer) occur more frequently than the rest because of the nature of the job in KFC kitchens. The classes are therefore imbalanced and the accuracy might not be the right measure. That is why we observe F1 weighted measure for multiclass case with imbalanced classes and it is represented in

column Table 2. Based on this measure, TRN performs the best on verb recognition, while MTRN performs the best on noun recognition task.

As TRN turns out to be the best model based on metrics, it would be interesting to see its distribution of the top 5 most probable output verbs/nouns. This is presented on Fig 2.

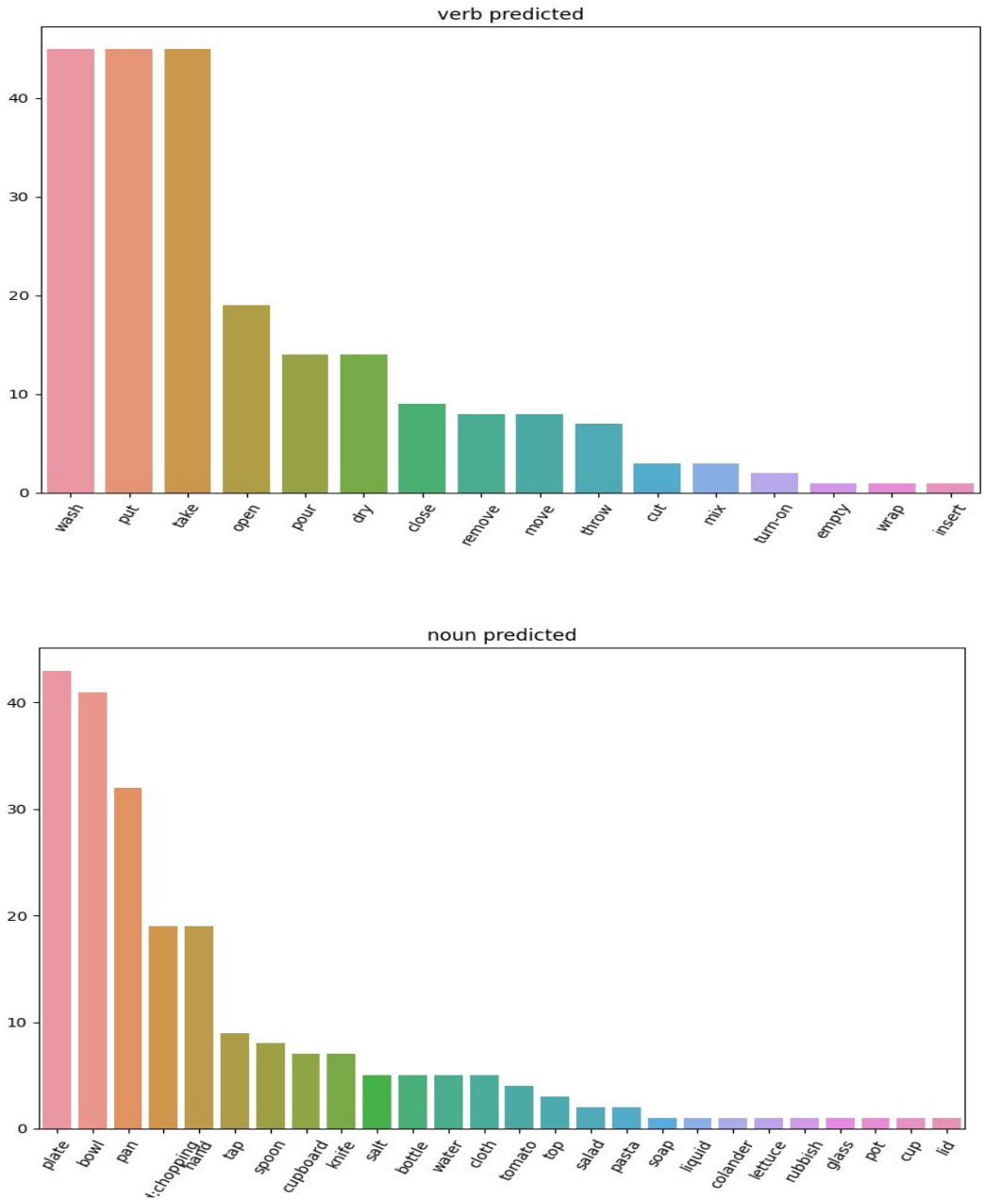

Fig 2: Distribution of most frequent predicted verbs/nouns from TRN model

Most frequent predicted verbs are wash, put, take, open, pour and some of them are related to labels, most frequent predicted nouns are plate, bowl, pan, board:chopping and it is clear that most of them are misses.

Conclusion

Labels for verbs and nouns (actions) in D1 dataset are taken arbitrarily based on a personal opinion. Dataset D1 does not include enough examples so that these results could converge, but still gives some light estimate of real-world scenarios and expectations. EPIC-KITCHENS pretrained models were trained on a comprehensive dataset of actions collected in different kitchens and this is the reason why it resulted in poor metrics when applied on KFC kitchens (with specific set of actions). However, these models are a great basis for further fine-tuning on specific cases such as KFC kitchens and applying to particular problems.

Task B

The goal of this task is to transfer learn one model on a custom dataset of KFC kitchen videos containing 3 verb classes (take, put, move) representing three different actions. However, after obtaining KFC videos from youtube it is noticed that one more verb is also very often, MIX, so this will be also added as a class and there will be 4 verb classes in total (take, put, move, mix).

Labeling process

Every video is split into video parts of the same length (number of frames) using FFMPEG program (***video.py***). These parts have 30% overlap (the percent is chosen arbitrarily). This is done so that some parts that contain incomplete or unfinished action could be found in the next (succeeding) or previous (preceding) video part and therefore would give much better representation of action. There are also some actions that last longer (for example, mixing flour) and each subsequent part has same label and contains last 30% of previous video part. The assumption is that this will give a variety of the same scene and action and thus, additionally augment data. Since there are just 7 KFC videos downloaded from the internet (avg. duration ~5min), there won't be many usable video parts, so overlap will increase their number providing reasonable variety between them. The biggest challenge when assigning labels was to differentiate between TAKE and MOVE actions, as well as between PUT and MOVE. Whenever some object is transferred from one place to other (or from one plate to other plate) this is called MOVE action. Whenever some object is put into some box (or fridge, cooker, pot), this is called PUT. TAKE is called when some object is taken from some place by hands or with some tool and holded. MIX is called everything that is related to shaking some object or mixing some mixture.

The whole process is conducted for 2 different video part lengths (60 and 120 frames).

Dataset D1 from Task 1 will be used as a **test** dataset and for comparison between original EPIC KITCHEN model and fine-tuned one. The set of actions present here surpass these 4 mentioned, so from D1 will be taken only those that are similar to these 4. This requires going through all of the labeled actions and replacing current verbs with one of these (take, mix, put, move) if the change doesn't affect the original meaning of the action. The new labels are given in *data/test/action_recognition_labels_test.csv* and this dataset will be called D3. Their distribution is presented in table 4:

Verb	Count
mix	9
put	9
take	6
move	3

Table 4: Count of verbs in test set D3

Transfer learning

In order to adapt the TSN model of the network for this purpose the last fully connected layer of shape 2048 x 125 is replaced with another of shape 2048 x 4. Since the dataset is small and similar to the original dataset used for pretraining the model, codes that extract meaningful information from the input will remain unchanged, while we will train only the last FC layer. Therefore, gradient descent will not have impact on previously trained parameters, except to the last FC layer of shape in_features x 4 (its weights and bias).
Data is fed to the model using `torch.utils.data.DataLoader` which is initialized with ***train_set*** (***val_set***) of type *ActionDataset* (class inheriting `torch.utils.data.Dataset`). Train and val size ratio is 70:30, batch size is 5. The loss criteria is cross entropy optimized with Adam optimizer.

Segment length (frames)	Train Loss	Val Loss	Num epochs	Train Acc@1	Val Acc@1	Test Acc@1	EPIC KITCHENS Acc@1
60	0.9594	1.1088	2	0.6171	0.5733	0.3703	0.3333
120	0.3551	0.4082	4	0.8905	**0.8667**	**0.407**	0.3333

Table 5: Results of transfer learned models for different segment lenghts
Figure 3 and 4 present training process and how the optimal number of epochs is found.

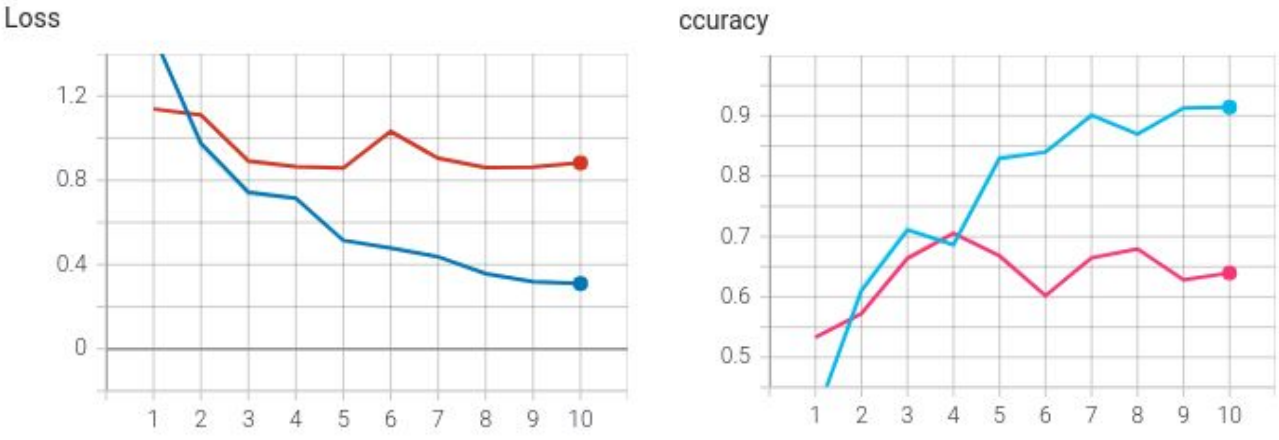

Figure 3: Train Validation Segment length (frames) = **60** learing_rate=0.00097[1]

[1] Figures 3 and 4 are generated using the tensorboard package.

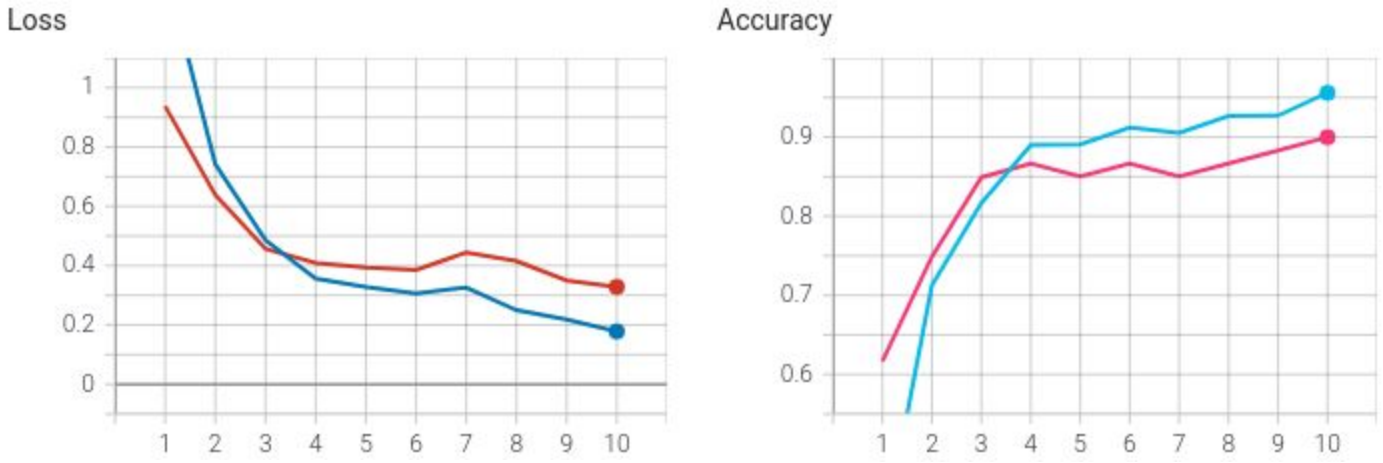

Figure 4: Train Validation Segment length (frames) = **120** learing_rate=0.00097

EPIC KITCHENS Acc@1 represents accuracy from inference on D3 using the original EPIC-KITCHEN TSN model.

For train process there is the script train.py with arguments:
python train.py --videos_dir data/video_segments/videos_60 --output_dir ckpt_60 --lr 0.00097 --num_epochs 10

For evaluation on fine-tuned model there is the script evaluate.py with arguments:
python evaluate.py --checkpoint_dir ckpt_60 --model_number 5

For evaluation on the original EPIC-KITCHEN model there is the script test.py with arguments:
python test.py --model 'TSN' --dataset 'D3' --output_dir output

video_segments folder which should be downloaded from and moved to data/ directory.
Hardware used is nVidia GeForce 1050Ti, 8GB RAM and i5-7400 CPU @ 3.00GHz.

Conclusion

We get better results when trained on longer segments (bigger number of frames). This may be the case because the action is more continual and there are stronger changes in movements which decides the nature of the action.
The accuracy and the loss could be improved by getting more variety in data through augmentations and data collection such as: different angles views (perspective changes, mirror, zoom), lighting conditions (brightness, contrast, saturation), different backgrounds (different places to record), distorting quality of video (adding noise, blurring, image compression).

Speech data could be used in two ways. One is to automatically label action by extraction verb and noun from audio. In this way, there won't be the need for manual labeling, except for some cases as supervision. The second is to be used as additional features concatenated to the existing CNN codes from video and forwarded to the FC network. This might improve the accuracy in cases when the person that is recorded is performing some action and the sound of the environment or person speech is related to the action.