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.

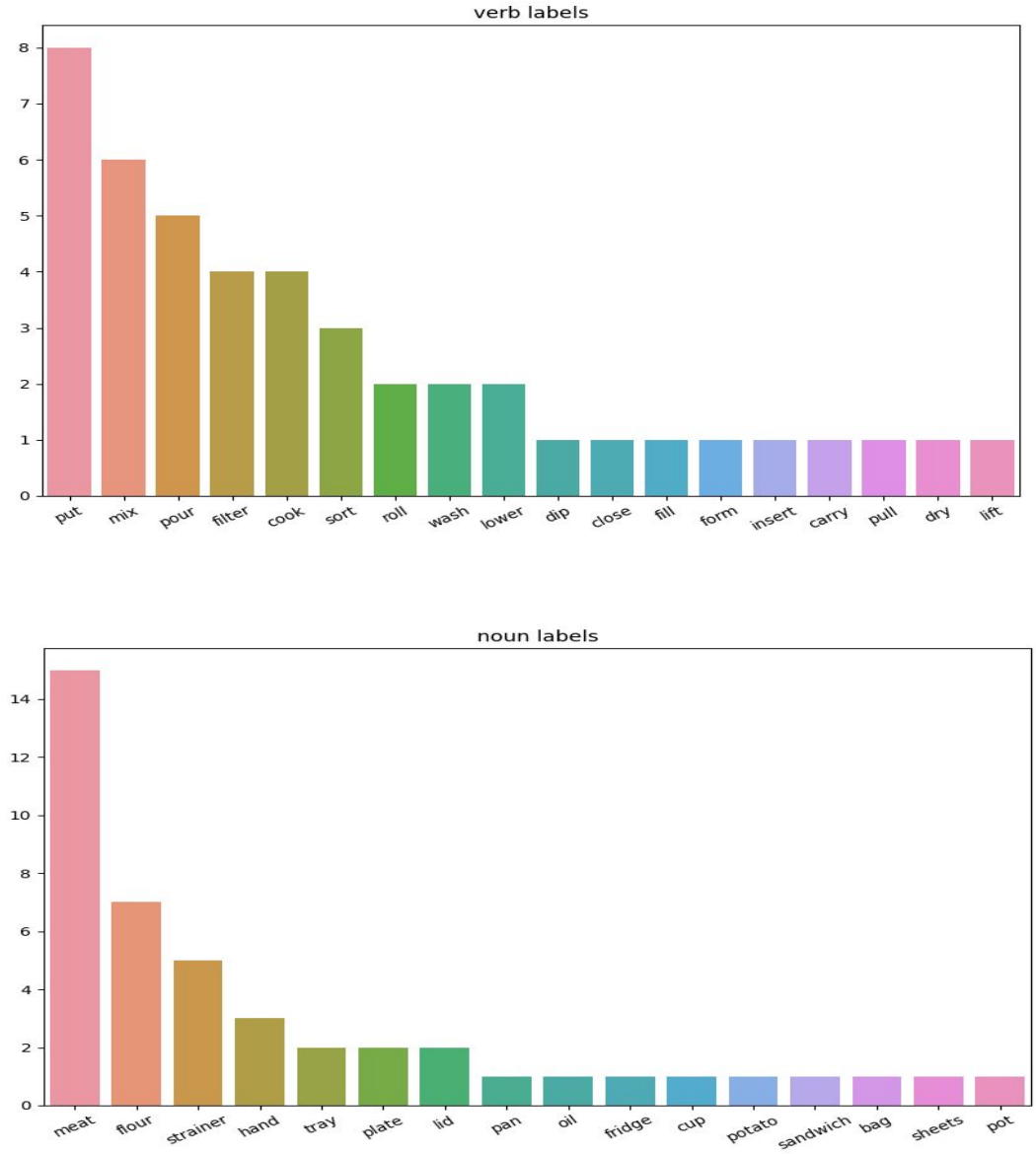

Fig 1: Distribution of most frequent verbs and nouns labels in test dataset D1

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.

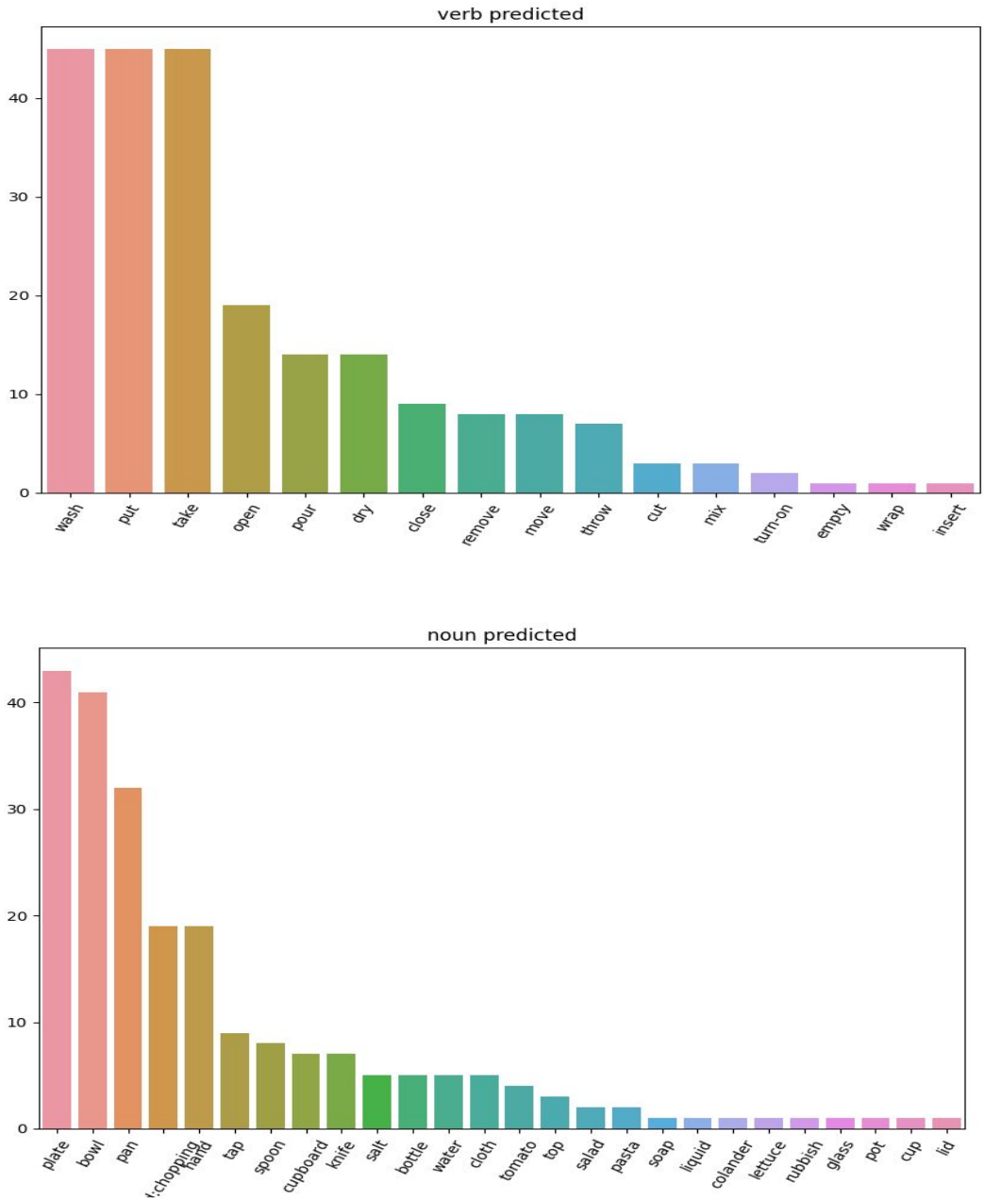

Fig 2: Distribution of most frequent predicted verbs/nouns from TRN model

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.