Detecting and Predicting Visual Affordances of Objects

By Bhumika Kaur Matharu
18 May, 2021

Advisor: Chris Pollett
Committee Members
Robert Chun
Sunhera Paul

Overview

- Objective of Project
- Introduction
- Background
- Model Design
- Dataset and Data Preprocessing
- Experiments and Results
- Conclusion and Future Work

Objective of Project

- Using computer vision techniques, implement a model to learn object affordances through RGB videos
- Detect and predict the affordance of an object along with the interaction region between human and object

Introduction

- Rapid growth of autonomous robots is transforming many industries (Manufacturing, Healthcare, Entertainment, Defense)
- Autonomous robots
  - Perform high risk tasks
  - Have low error rate
  - High efficiency
  - High speed and reliability

Existing Problems

- Robots can perform best under controlled environments, making them unreliable for real world scenarios
- Slightest change in the environment or an encounter with novel object requires human supervision and re-training the robots
- Difficult to comprehend and perform suitable actions on a new object in the environment
- One of the ways to overcome these challenges is **Affordance learning**

Affordance

- Affordance tells what actions a user can perform on an object in given surroundings
- Affordance learning enable robot to learn and discover set of possible actions on a novel object
- Allow robot to perform human-like actions without human supervision.

Background

- Following neural networks and models were used in the project
  - Convolutional neural networks
  - Recurrent neural networks (LSTM and ConvLSTM)
  - Fully convolutional network
  - Transposed convolution
  - Attention model

Convolutional Neural Network (CNN)

- Used to analyze and extract features from any kind of image or video data
- Consists of convolutional layers, activation layers, max-pooling layers and fully convoluted layers

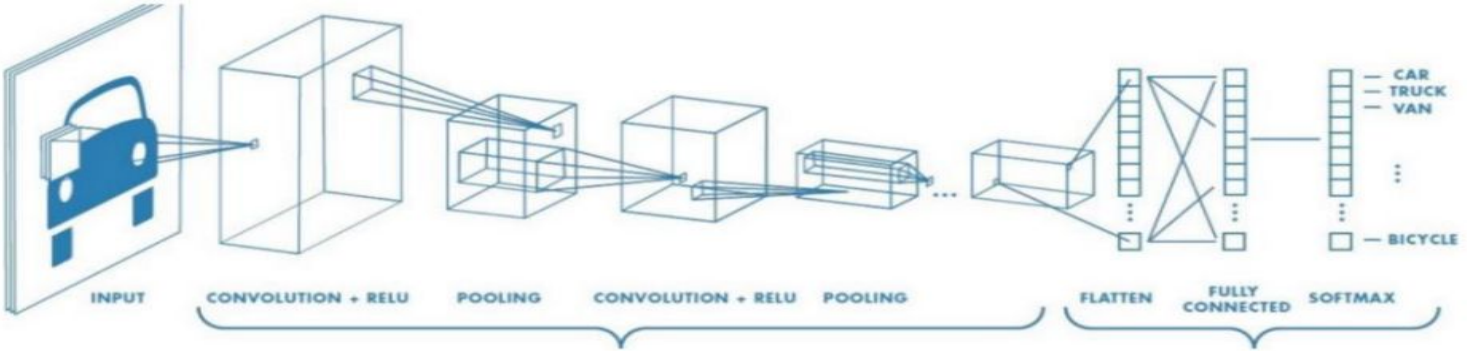


Fig 1

Recurrent Neural Networks (RNN)

- Used for sequential data or time-series data
- Remember previous input while producing the output
- Output is influenced by hidden state representing prior input/output

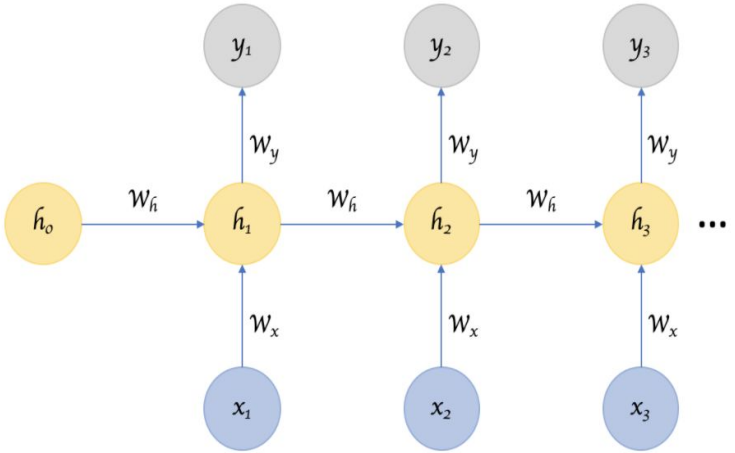


Fig 2

Long Short Term Memory (LSTM)

- Vanilla RNN can’t handle long input sequence
- Traditional RNN suffers from exploding gradient problem
- Long Term Short Memory units (LSTM) minimize the problem through gates
    - Update gate
    - Relevance gate
    - Forget gate
    - Output gate

Convolutional LSTM (ConvLSTM)

- LSTM fully connected layers does not encode any spatial information in case of spatiotemporal data
- ConvLSTM overcome the problem by using 3D tensors, where last two dimensions i.e. rows and columns are spatial dimensions
- Captures underlying spatial information by performing convolution operation at each gate in LSTM cell instead of internal matrix manipulation

Fully Convolutional Network (FCN)

- FCN performs classification at per-pixel level on an input image
- Consists of fully convolutional layers
- Outputs the category prediction of each pixel corresponding to spatial position

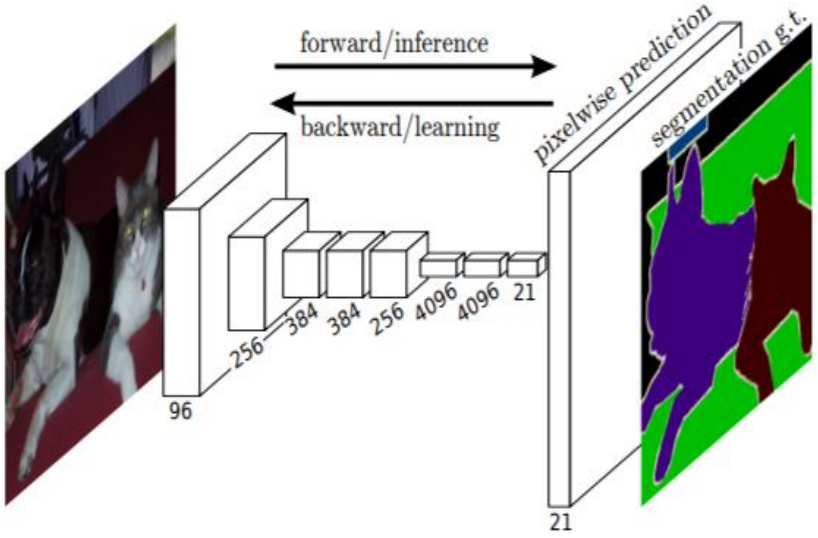


Fig 3

Transposed Convolution

- Often mistook as deconvolution operation
- Combine two operations: upscaling of an image with convolution
- Upsamples the output feature map to the input size to predict values at each pixel
- Performs normal convolution operation in the opposite direction

Attention Model

- Tells the network which relevant part to focus on in a sequence or visual input.
- Gives different score to each weight which represents the relevance
- Types of attention:
  - Soft Attention
  - Hard Attention

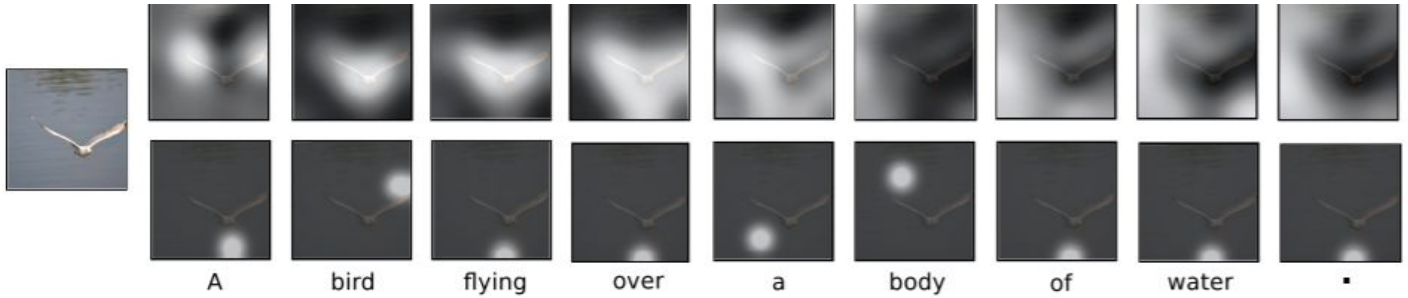
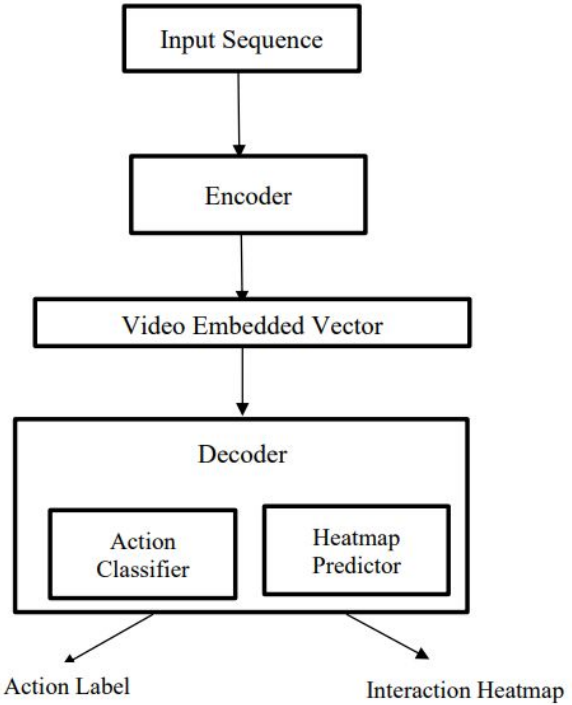


Fig 4

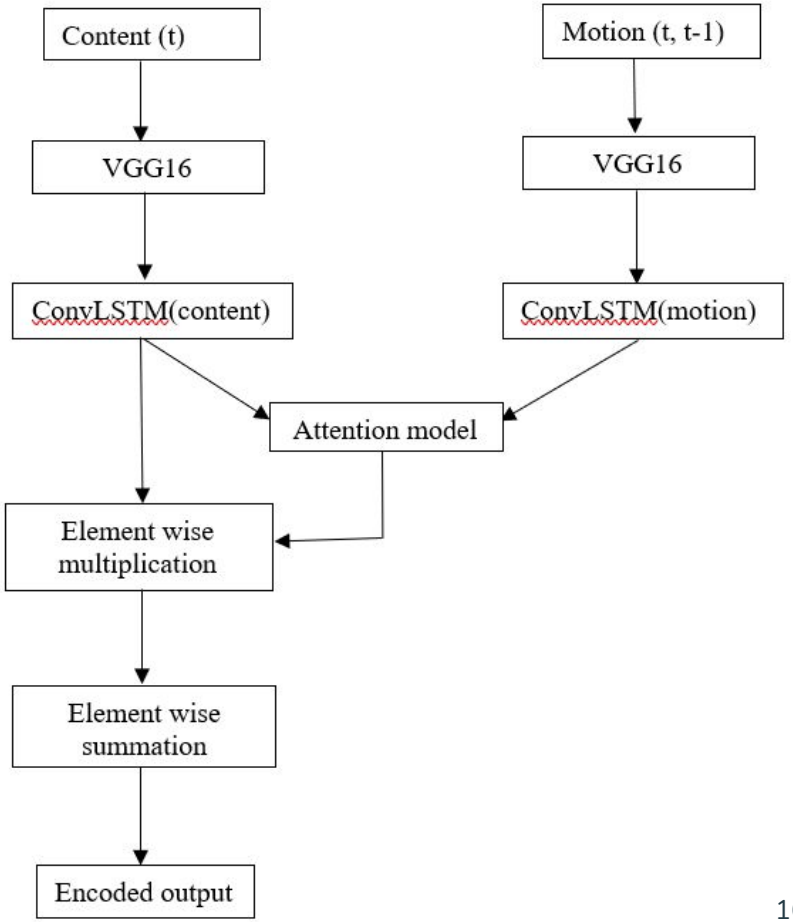
Model Design

- Model is based on an encoder-decoder architecture.
- Encoder takes the frames of the video as the input and produces an embedded vector of each video.
- Decoder is composed of two models: action classifier and heatmap predictor
- Both of the tasks are trained in a multi-task (jointly) manner



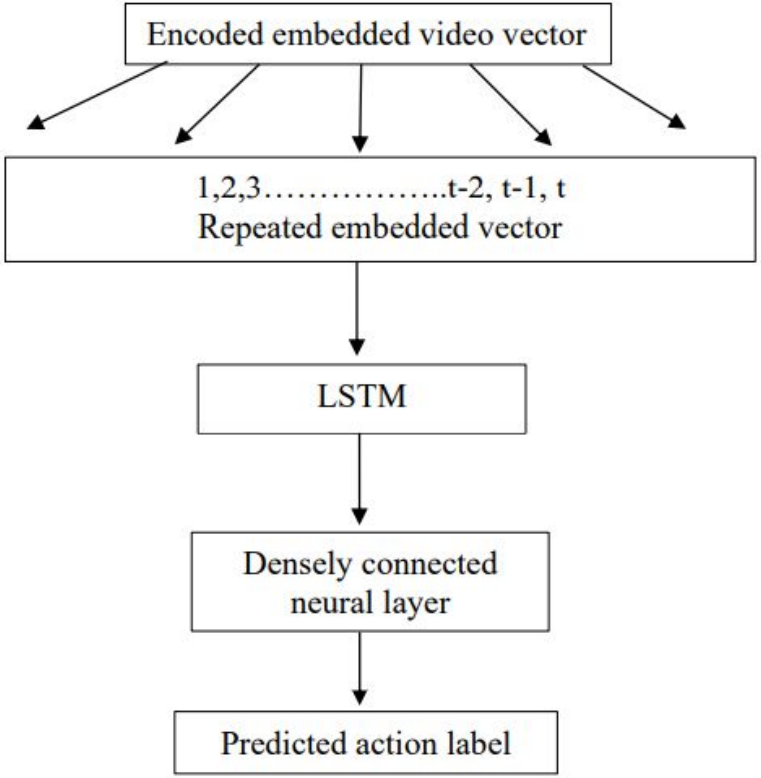
Encoder

- Encoder takes two inputs: motion and content of the video.
- Visual features extracted from from pre-trained VGG16 is fed into ConvLSTM
- Attention mechanism is used to aggregate the output of ConvLSTMs per time step
- Output of the encoder is a low dimensional embedded feature vector extracted from the video.



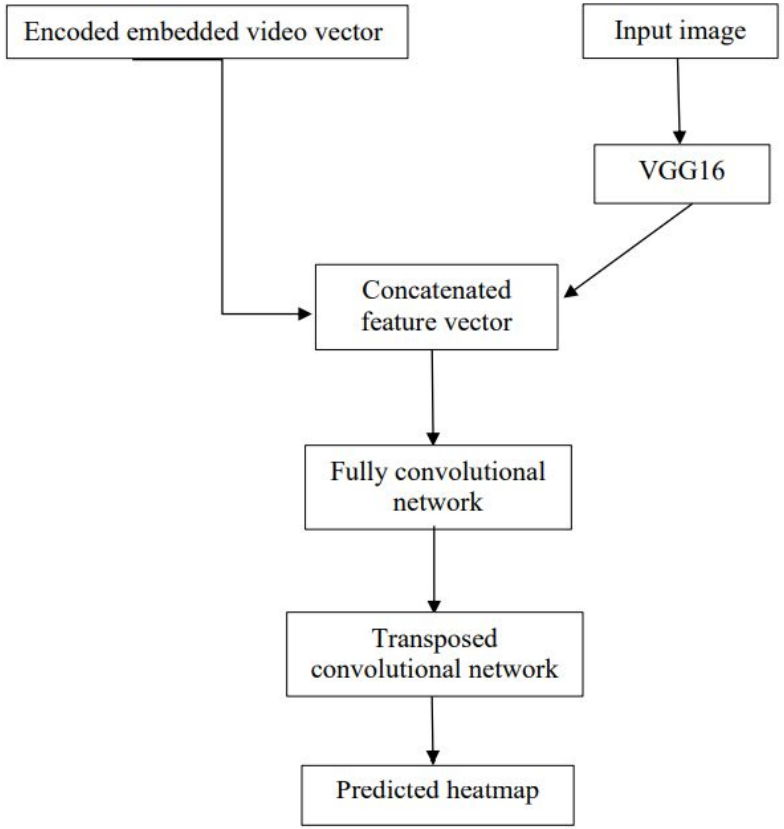
Action Classifier

- Encoded output is taken as input
- Input is repeated as many times as the number of time steps in a video
- LSTM is used to predict the affordance of the given task

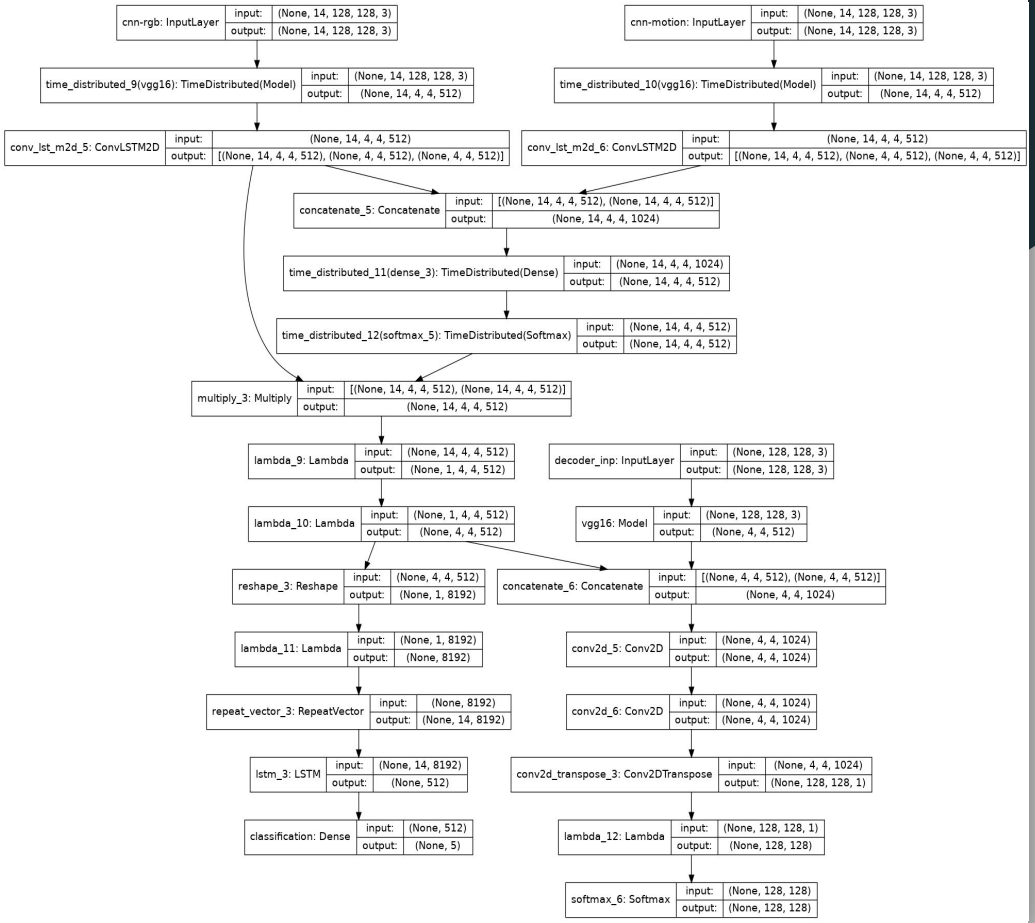


Heatmap Predictor

- Two inputs: Encoder output and object image
- Fully convolutional network used to predict the interaction region
- Transposed convolution to upscale the result to the input size



Model Implementation

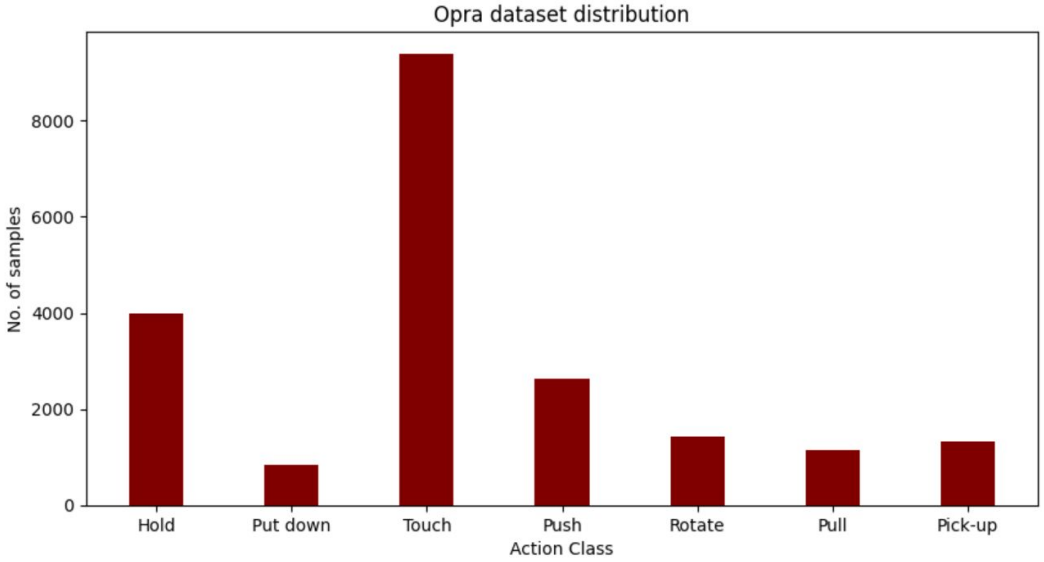


Dataset

- Online Product Review dataset for Affordance (OPRA) is used
- Consists of 11,505 third person demonstration videos and 2,512 object images collected from various YouTube product review channels.
- Affordance information of a task is incorporated by annotating 10 points on the object image highlighting the human-object interaction region.

OPRA Dataset Distribution

- Consist of seven action classes
- Total of 20,774 video clips are collected
- 16,976 videos for training and 3,798 for testing



Data Preprocessing

- OpenCV and FFmpeg are used for data pre-processing
- The videos are segmented into smaller videos, where each video contains some human-object interaction
- Content Frames: For each segmented clip, 15 frames are generated at a 1 fps
- Motion Frames: Absolute differences between consecutive frames (frame at t and t-1 time step)
- Custom data generator created to feed multiple input at once
- Gaussian blur ground truth heatmap computed using the annotated points on the object image

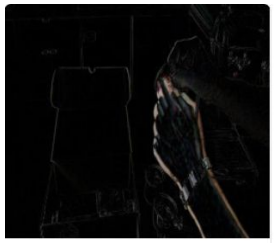

Capresso
OFF / ON
INSERT
LOCK

Motion and Content Frames

2_200.3_3.6f_0001.jp...

2_200.3_3.6f_0002.jp...

2_200.3_3.6f_0003.jp...



2_200.3_3.6d5.png

2_200.3_3.6d6.png

2_200.3_3.6d7.png

Experiments and Results

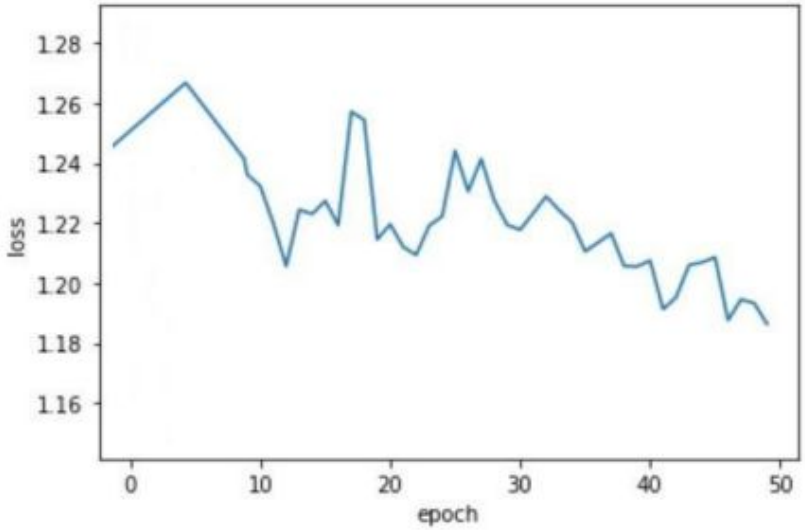
Model Parameters

- Adam optimizer
- Learning rate: 0.0001
- Action classifier loss: Categorical Cross-entropy
- Heatmap Predictor loss: KL divergence

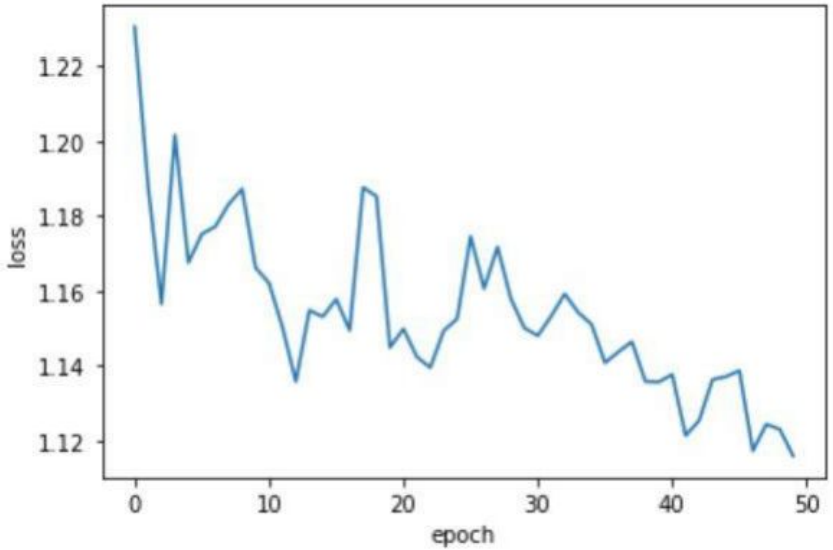
Experiment 1

- Run the model on three classes :
  - Hold with 730 videos
  - Touch with 800 videos
  - Push with 775 videos
- Iterations: 50 epochs
- Batch size: 12
- Input frame and image size: 128 by 128

Results - Loss



Overall Loss

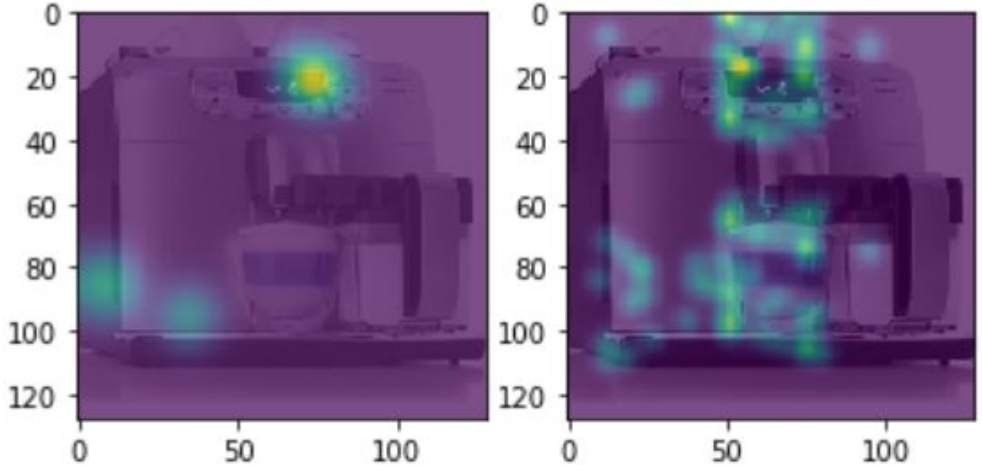


Classification Loss

Results - Prediction

```
print(prediction[0])
ind = np.argmax(prediction[0], axis=1)
print(action_labels[ind[0]])
```

```
[[0.32215092 0.3157147  0.36213437]]
push
```



Ground Truth: Push

Statistics of the Experiment:

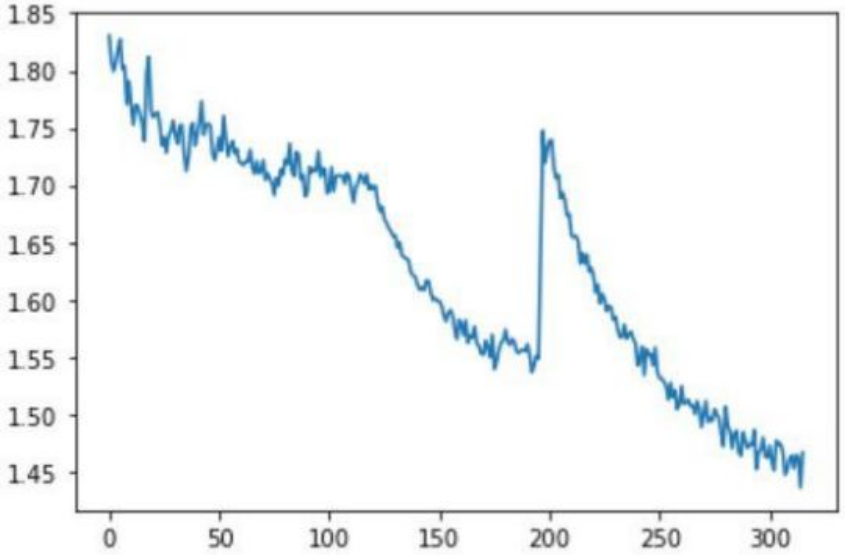
Classification Accuracy: 35.7 %

KL Divergence Loss: 6.97

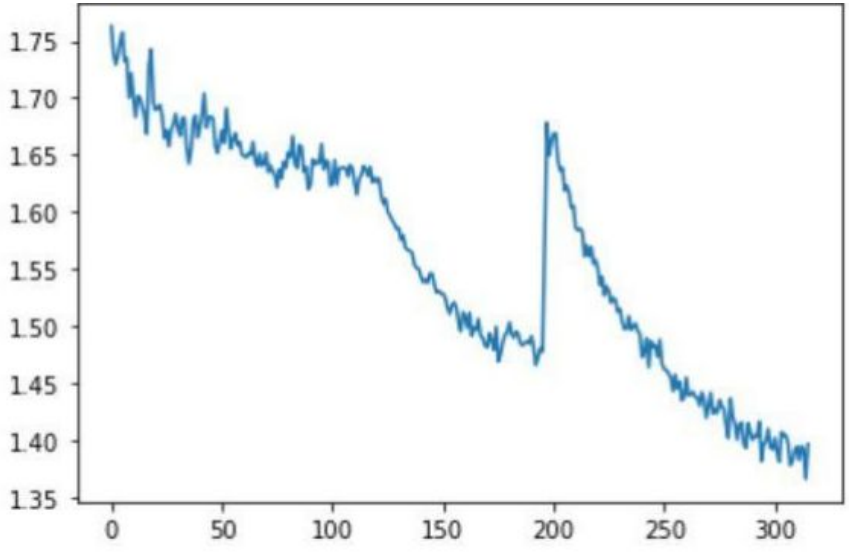
Experiment 2

- Run the model on five classes :
  - Hold with 415 videos
  - Touch with 410 videos
  - Rotate with 408 videos
  - Push with 412 videos
  - Pick-up with 365 videos
- Iterations: 315 epochs
- Batch size: 12
- Input frame and image size: 128 by 128

Results - Loss



Overall Loss
x-axis: epoch, y-axis: loss

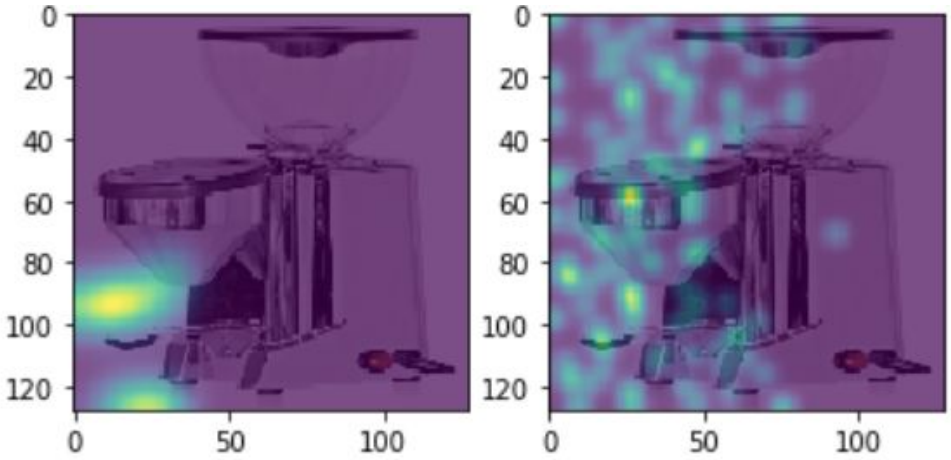


Classification Loss
x-axis: epoch, y-axis: loss

Results - Prediction

```
print(prediction[0])
ind = np.argmax(prediction[0], axis=1)
print(action_labels[ind[0]])
```

```
[[0.13899542 0.3731257  0.3403442  0.05909514 0.08843959]]
touch
```



Ground Truth: Touch

Statistics of the Experiment:

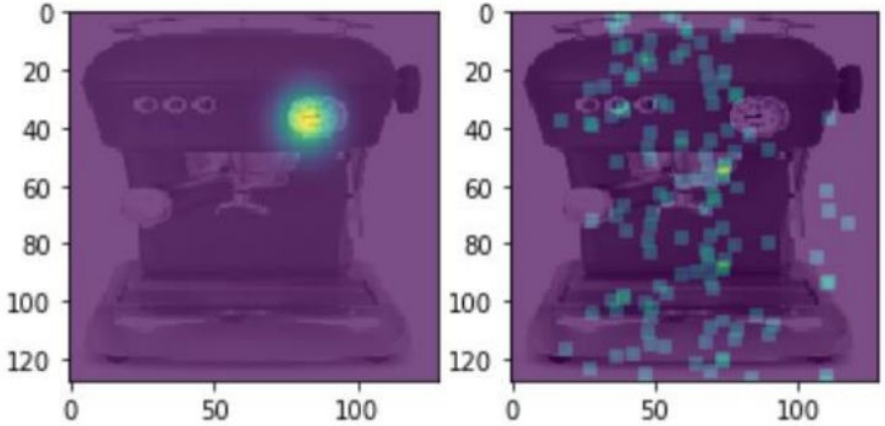
Classification Accuracy: 48.48 %

KL Divergence Loss: 7.012

Results - Failures

```
[100] print(prediction[0])
      ind = np.argmax(prediction[0], axis=1)
      print(action_labels[ind[0]])

      [[0.2202711  0.48487148 0.29485744]]
      touch
```



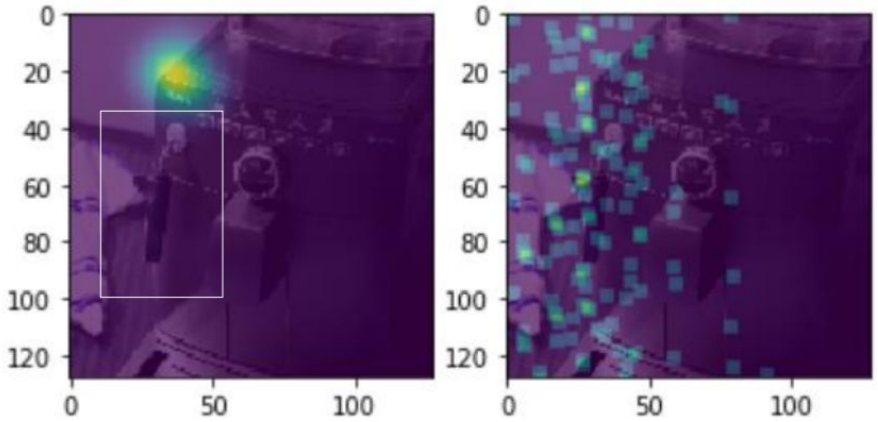
Ground Truth: Push

Ground Truth: Touch

Distortion in Dataset

```
[ ]  action_labels = {0: 'hold', 1: 'touch', 2: 'rotate', 3: 'push',
     print(prediction[0])
     ind = np.argmax(prediction[0], axis=1)
     print(action_labels[ind[0]])
```

```
[[0.08775322 0.19458954 0.4061065  0.25111073 0.06044002]]
rotate
```



Conclusion

- Achieved the objective of detecting and predicting affordance of the object and interaction heatmap between user and object.
- Accuracy of the action (affordance) classifier increases with the increase in the duration of training.
- The heatmap predictions are better with more number data samples per class.
- The action classifier and heatmap predictor trained in a multi-task manner, taking advantage of the joint learning.

Future Work

- Accuracy of the heatmap predictions can be increased by more number of data samples for each action class
- Custom loss function for better training
- Different hyper-parameters (learning rate)
- Different pre-trained models and weights

References

Fig 1 - M. Taylor, “Computer Vision with Convolutional Neural Networks”, https://medium.com/swlh/computer-vision-with-convolutional-neural-networks22f06360cac9#:~:text=Most%20computer%20vision%20algorithms%20use,and%20edges%2C%20from%20spatial%20data.

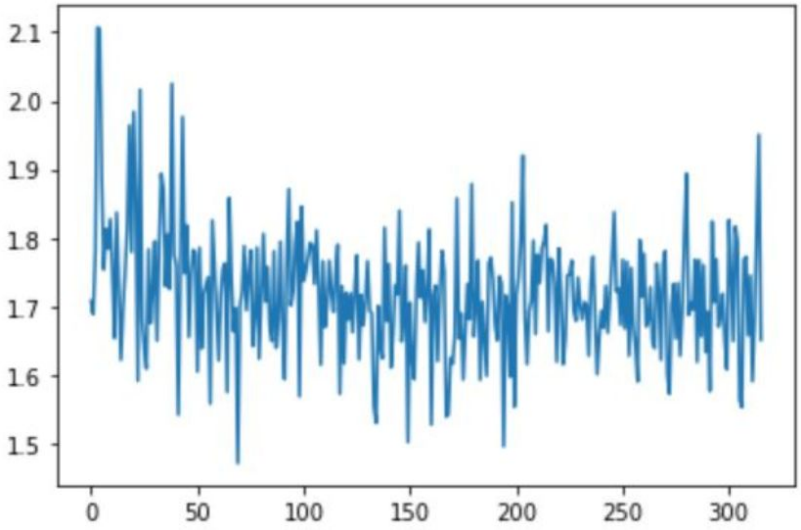
Fig 2 - M. Venkatachalam, “Recurrent Neural Network”, https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Fig 3 - J. Long, E. Shelhamer and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440

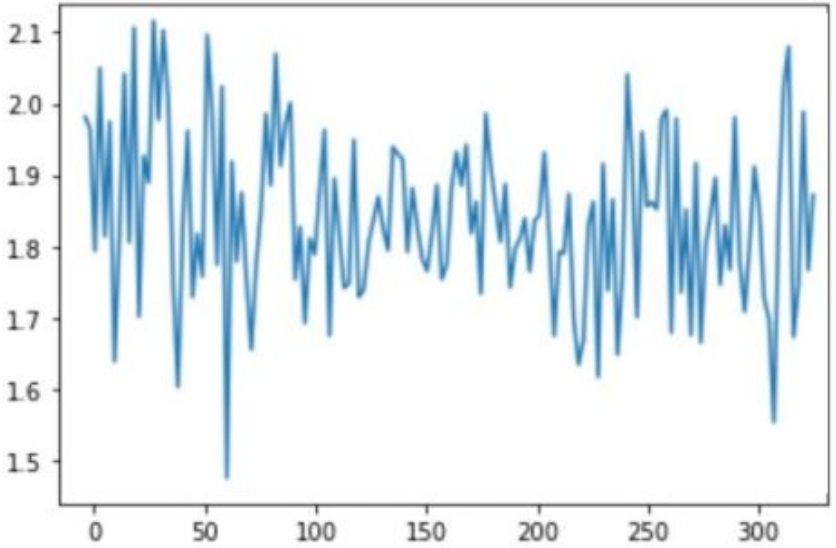
Fig 4 - K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel and Y. Bengio, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, in Proceedings of the 32nd International Conference on Machine Learning, 2015.

Thankyou!
Any Questions?

Additional Experiment



Validation Loss - Together
x-axis: epoch, y-axis: loss

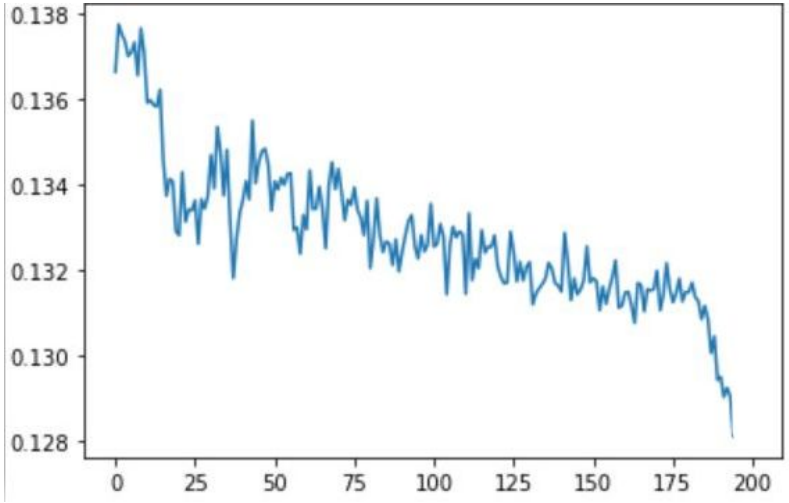


Validation Loss - Individual
x-axis: epoch, y-axis: loss

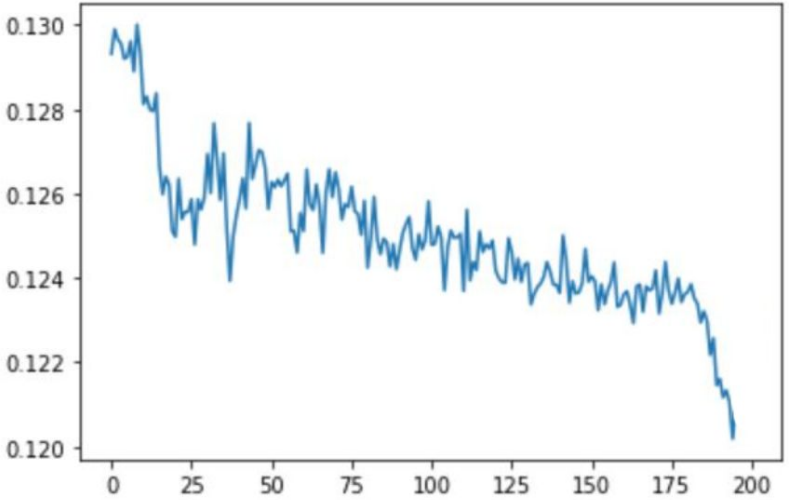
Additional Experiment

- Run the model on seven classes :
  - Hold with 180 videos
  - Touch with 200 videos
  - Rotate with 201 videos
  - Push with 190 videos
  - Pull with 170 videos
  - Pick up with 180 videos
  - Put Down with 195 videos
- Iterations: 200 epochs
- Batch size: 12
- Input frame and image size: 128 by 128

Results - Loss



Overall Loss

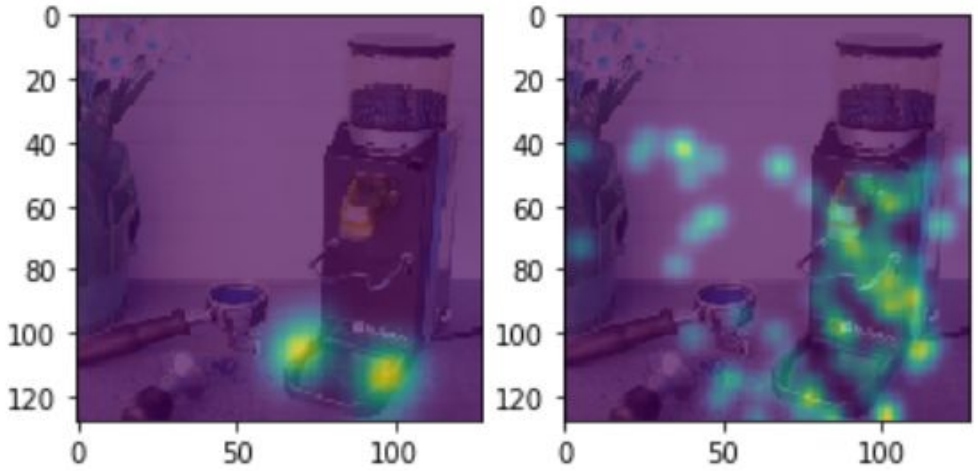


Classification Loss

Results - Prediction

```
print(prediction[0])
ind = np.argmax(prediction[0], axis=1)
print(action_labels[ind[0]])
```

```
[0.15427716 0.1333073  0.18406256 0.1200726  0.08894752 0.18849538
 0.13083749]
pick_up
```



Ground Truth: Pick up

Statistics of the Experiment:

Classification Accuracy: 33.7 %

KL Divergence Loss: 7.07