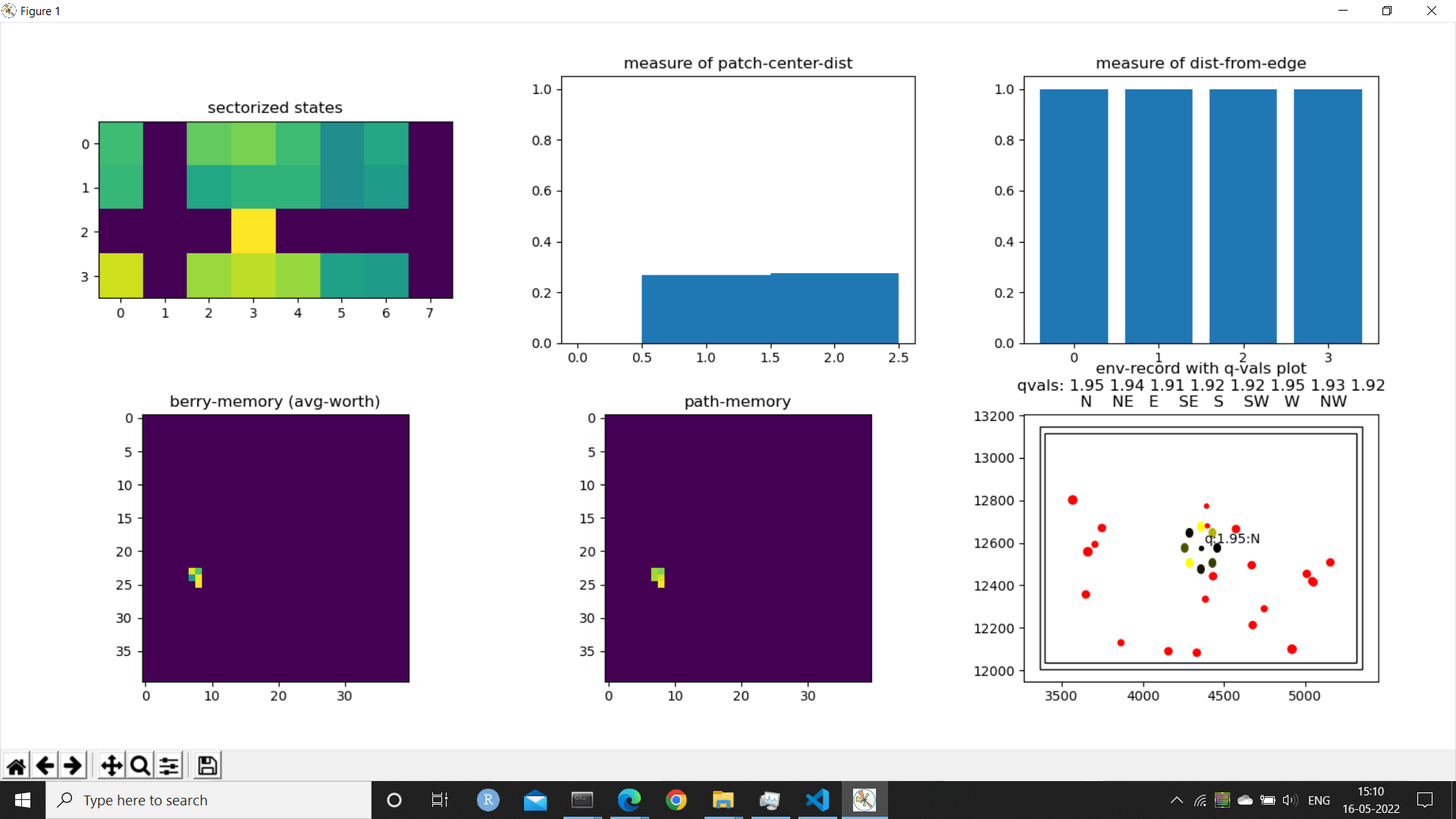
Added memories in form of matrices & scalar – each cell represented a portion of the berryField.

1. Berry-memory matrix: each cell contained the average worth of the observation when the agent was in the corresponding portion, the cell is updated using exponential averaging, as the agent passes through the portion, the cell’s value becomes a somewhat better estimate.

Also, each a cell gets updated whenever the agent gets into the corresponding portion of field.

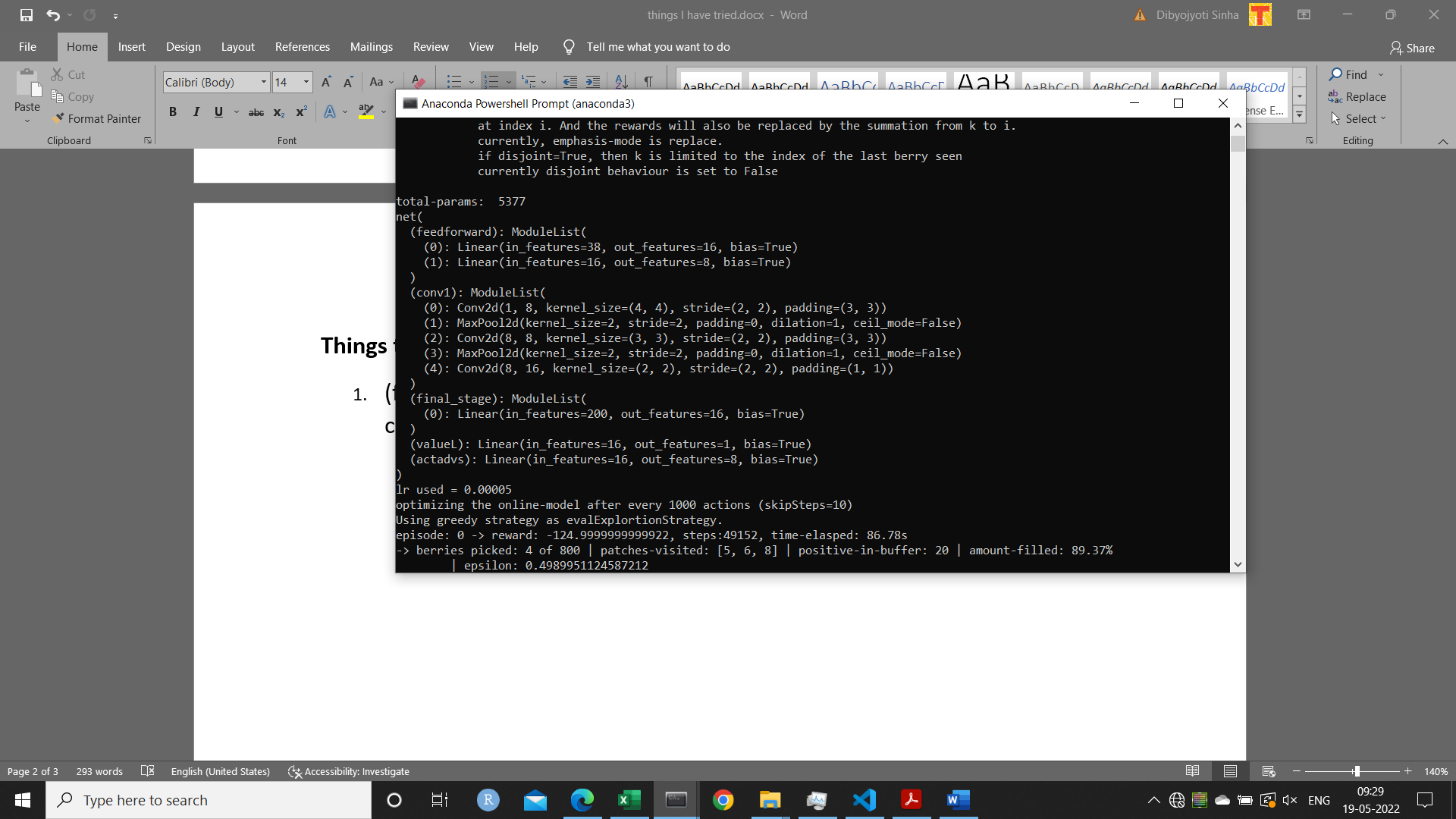
1. Path-memory-matrix: again, each cell represents a portion of the field. For every step (i.e., the end of skip\_steps when makeState is called) the entire memory is decayed by a multiplier. A cell’s value is made 1.0 if the agent is currently in the corresponding portion.
2. Time-memory: This is a scalar, but uses a matrix to store the data. The corresponding cell of the portion of the field the agent is currently in is incremented by a delta. Also, for every step (i.e., the end of skip\_steps when makeState is called) the entire memory is decayed by a factor of 1-delta.

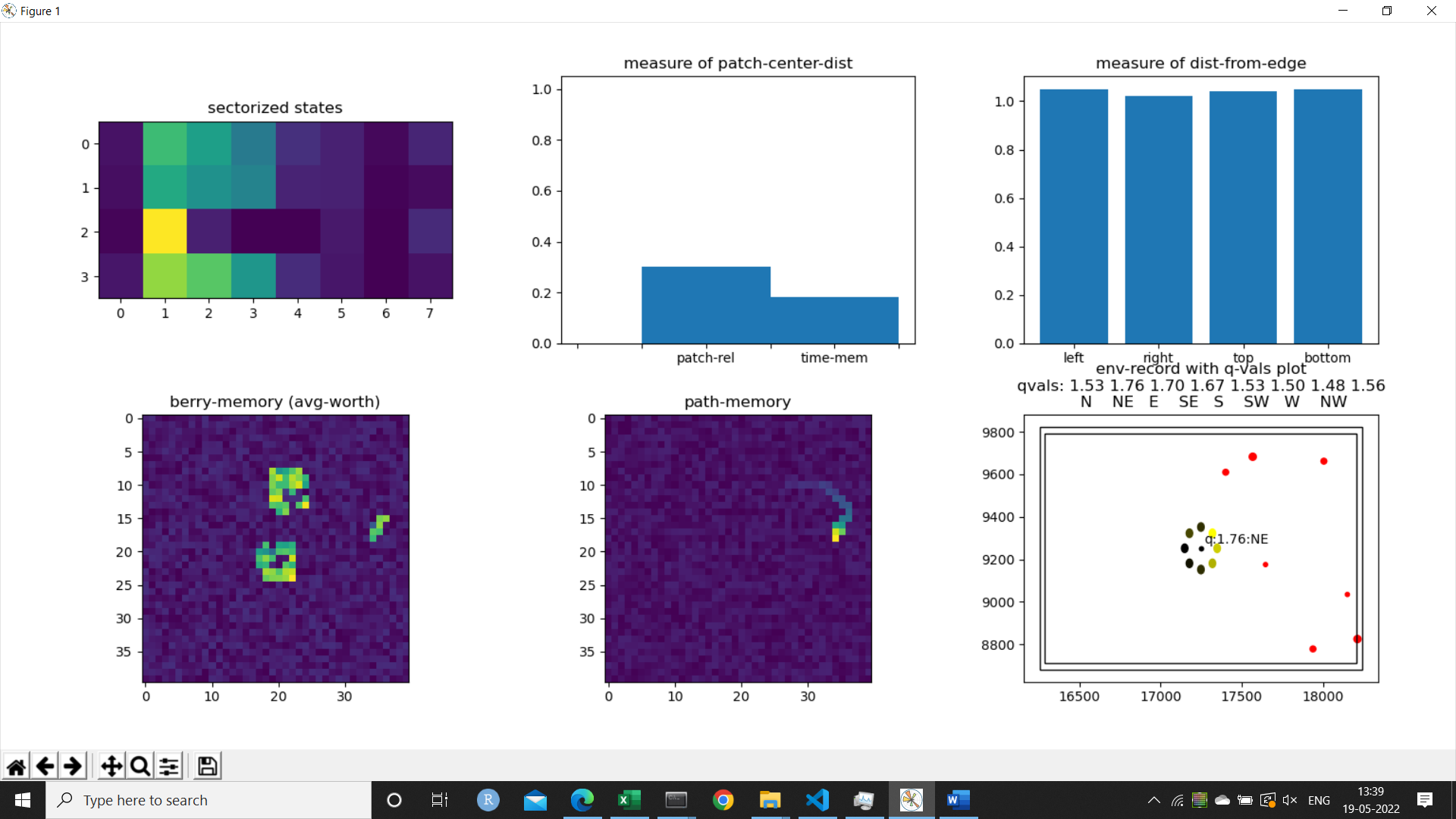


Time-memory

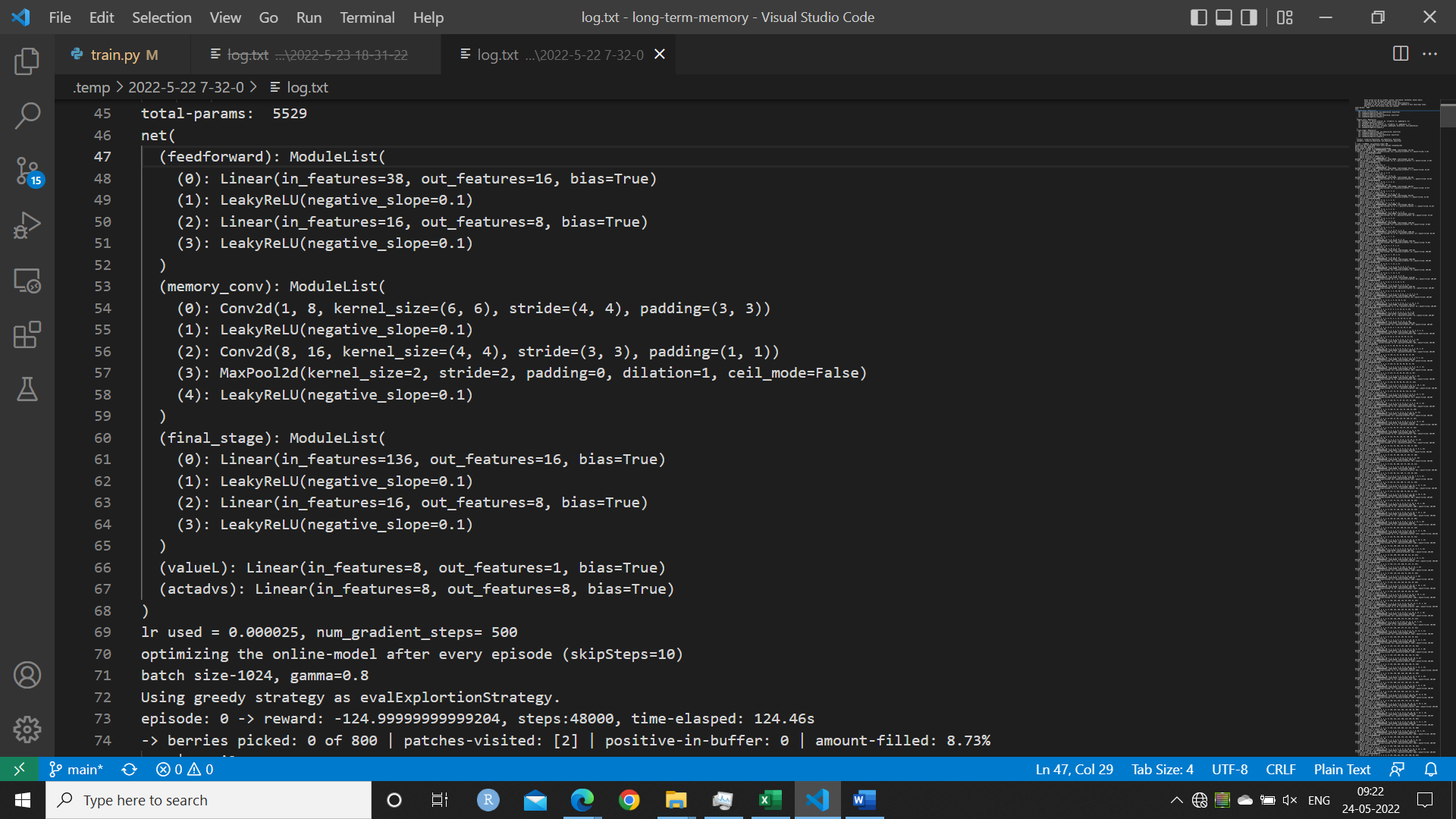
**Things that kind-of worked:**

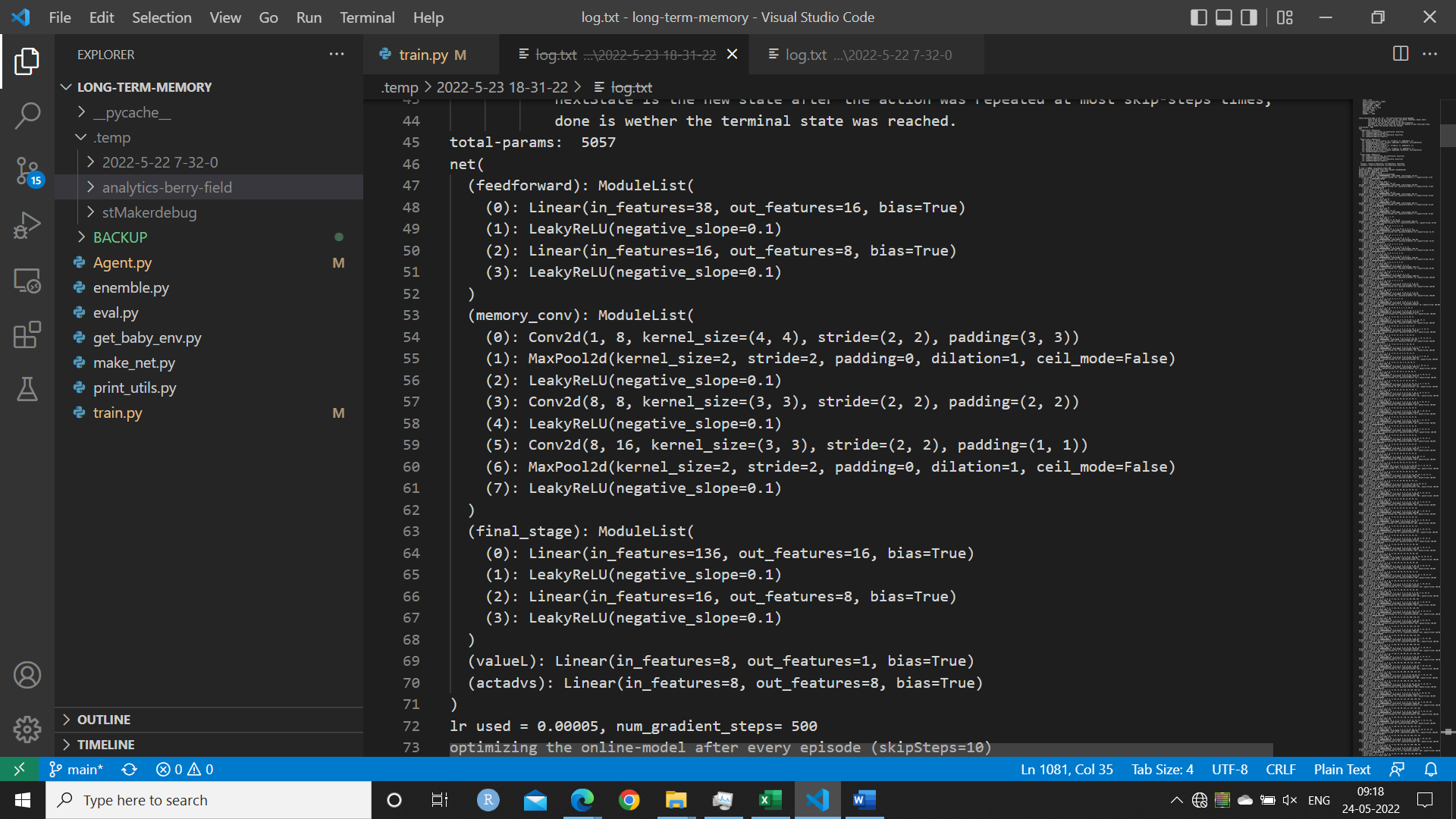
1. (folder-‘3’) Using the same conv net on both memories (basically concatenate the memories into a single array of size (80,40). And reduced gamma to 0.8 and lr to 0.00005. This collected 296 of 800 berries in episode 312.





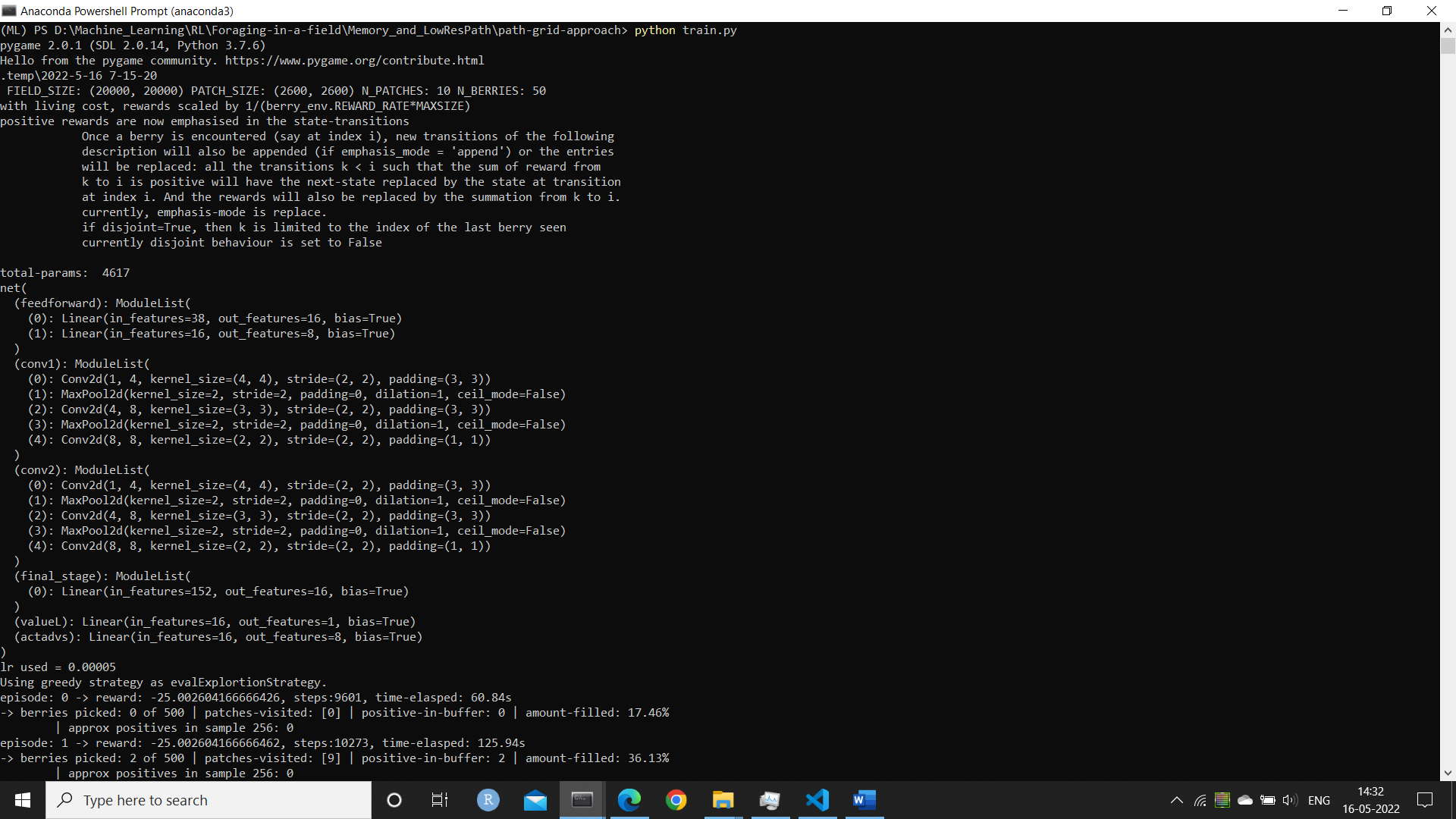
1. (folder-‘5’) A 2-layer shallow conv-net seems to adhere to exploitation and retirement at the edge when trained only at the end of the episode with 500 gradient steps, learning rate = 0.000025, batch size = 1024, polyak-tau = 0.2, update-frequency = 5, priority-alpha = 0.99, and state-transition-mode = ‘single’. The original training files are in folder-‘5’. It occasionally reaches 100+ (and once in a while 200+) but then fall down to 30 – 80 range. However this learned much faster than the model in folder-‘3’. The training is still going on, will update this entry later.



1. (folder-‘6’) a similar architecture to folder-‘3’. But trained in a similar way to folder-‘5’ the only difference being that update-freq was se to 10, and learning-rate = 0.00005 was used. This was only trained till episode 168. This also exhibited behaviour similar to the model of folder-‘5’ in its similar age of training – to half exploit a patch (maybe it does-not remember that there are large berries still left) and mode to a fixed direction (north in this case) to ‘explore’ it may again half exploit a patch if encountered, then eventually get stuck till death at an edge.  
   Maybe more gradient updates are required. Folder- ‘3’ had updated its model every 1000 actions for 200 gradient steps. But for folder-3 the state-transition-mode = ‘all’ was used.  
   

**Things That did-not Work:**

1. Flattening the entire state and feeding into a fc network: Too many learnable parameters.
2. (Folder named ‘2’) A fully-connected-feedforward for the sectorized-states + edge-distances + patch-relative + time-memory. Separate conv-layers for each of berry-memory and patch memory. All were concatenated and fed into a dueling-net type of network. The conv-layer outputs were of shape (8,3,3). Total parameters are 4617. Tried multiple h-params with no avail.



My suspicions:

* 1. Perhaps the conv-layers were separated for far too long that the information from the different parts of the states could not be efficiently combined.
  2. Too high gamma?

1. (Folder named ‘4’) To reduce the bias in direction, I decided to use a 1d conv on sector-states. I tried two structures, both of them failed:
   1. Structure-1:   
      sector-state=(4,8) -> circular-pad(1) -> padded=(4,10) -> conv1d(kernel=3, ch=2) -> (2,8) -> circular-pad(1) -> (2,10) -> conv1d(kernel=3, ch=1) -> (1,8) conv-output
   2. Structure-2:  
      sector-state=(4,8) -> conv1d(kernel=2,stride=2, ch=2) -> (2,4) -> conv1d(kernel=2, stride=2, ch=4) -> (4,2) -> conv1d(kernel=2, stride=2, ch=8) -> (8,1) conv-output

In both the structures the output was flattened and concatenated with the memory-conv-outputs and fed into the final dueling layer.

1. Both memories as channel (2,40,40) input. This was too difficult to learn using the current algorithm and the default reward signals.
2. Tried to run a similar model as in Folder-3 with state\_transition\_mode = ‘single’ (in this mode only the [initial-state, action, reward, next-state] is appended where next-state is the state reached after the skip-steps), and the model was to be optimized only at the end of the episode. This was learning too slowly

**Things to try:**

1. Both memories as channel (2,40,40) input.