2)  How are you defining your loss function when training your model?

1. model.compile( optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'] )
2. we defined our optimizer as the Adam optimization algorithm, which is an optimizer that can be used to update the networks weights iterative based on training data
   1. adam is different from the classical stochastic gradient descent (maintains a single learning rate for all weight updates and it does not change during traing). Adam instead of only adapting the parameter learning rate based on the average first moment, it also adapts the average of the second moments of the gradients (uncentered variance)
3. we set out loss function as categorical cross entropy. This loss function is suitable for multi-class classification models where there are two or more output labels, which in our case we have 4 different labels for the 4 different directions in the two agents. The output label would be assigned as one-hot category encoding value which is the form of 0 and 1s. This output label (if as integer form) is converted into categorical encoding using keras.
4. Lastly the metrics we used is ‘accuracy,’ which according to keras it would be converted to BinaryAccuracy, CategoricalAccuracy, and SparseCategoricalAccuracy. The accuracy would be converted based on the loss function and the model output shape. Thus in our case the accuracy would be converted to CategoricalAccuracy.

3)  In training, how many episodes on how many different gridworlds were necessary to get good performance of your model on the training data?

* 1. For the first agent, we first flattened out data input of dim x dim
  2. Used two dense layers with units of 32 and 16.

4)  How did you avoid overfitting? Since you want the ML agent to mimic the original agent, should you avoid overfitting?

5)  How did you explore the architecture space, and test the different possibilities to find the best architecture?