#ret

On the exploratory notebook, I agree with your answers on exercise 0, although I think you could be more clear on your answer to problem 2. Why do repeated values imply optimal values? Are the values returned by image 2 actually optimal? You successfully found two images that produce positive values on exercise 1, and you also made new filters that behaves as you predicted, producing a positive value on the gradient image. I suspect, though, that this is not quite what you were trying to do — it might be interesting to come back to this one and think more about how you would detect a gradient. On the tuning exercise, you successfully increased accuracy to 81%, and I liked that you included all your code for both the models that didn't work that well, as well as the ones that did. You provided a brief summary of your results as to what worked better or worse, but I would challenge you to think more deeply about this: why do you think these are true? Are there experiments you could run that would prove some of your suppositions to be true? For example, you say that more convolutional layers might make things worse because we only have two epochs. How would you test to see if this is in fact the reason?

As for problem 2 with the repeated values, intuitively speaking, given that each group of values is repeating some number as opposed to repeating the same collection of different numbers their must be one optimal group which repeats the highest number. As for the gradients, without either large kernels or some type of memory or hidden cell state, I don't know how to make filters that would check for consistent gradients between sections. I would love to know the answer to this.

To test my hypothesis for my model, I could always run different types of models for more than 2 epochs and see how they perform. Less layers might also work better because of overfitting, or simply because the features we are trying to pick up aren't that detailed. Having larger filters then smaller filters makes sense for this problem, because we want to look for the larger features initially before processing them further with the smaller filters.

Good work, Huxley. I agree with all of your answers. Here are a few additional questions for your consideration: 1-2. Would there be any negative rewards? 1-3. If your input is a single 2d grid, how does the game know whether a given object is moving left or right? That is, if the current state contains a row that is car frog, how does the game distinguish between the car about to hit the frog, and the car passing by harmlessly? 1-5. I like this idea. How do you know what the maximum possible score is for a given state? 2-6. Tell me more about data harvesting. What ethical concerns does this present? 3-3. I think this will work fine, though I wonder: why only normalize rating and not other interval inputs, such as number of reviews? 3-4. I assume the F in FNN stands for fully-connected? If so, that will work, but is there something simpler you could try first? 4-5. Any reason why you picked f-score instead of precision, recall, or accuracy? 4-6. Any privacy issues here?

For 1-2, there could be negative rewards for negative actions, like the frog dying or hitting some kind of obstacle. In 1-3, we could either have the RNN handle it, or pass in extra states ourselves like directionality or velocity. 1-5, I am not to knowledgeable on Frogger mechanics but I do believe it is a completable game. Thus, we can find the maximum scores. We could also set a point goal for our agent to achieve, and use that as our maximum. 3-3, those other inputs should most likely be normalized as well. 3-4, a type of simple regression model could work as well. 4-5, accuracy is used when true positives/negatives are valued more than false positives/negatives, which is not what we want in this scenario. F-score combines recall and precision