Resources:

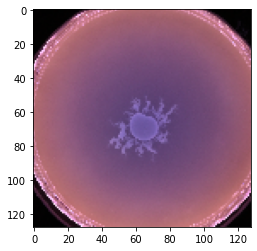
Similar code for mnist: <https://keras.io/examples/vision/conv_lstm/>

Custom dataset implementation: [Next-frame prediction with Conv-LSTM - Keras Code Examples](https://www.youtube.com/watch?v=P5yv8HDFc_M&ab_channel=HenryAILabs)

How to run:

You can run the code linearly cell wise if this is the first time running this code. But when running it for the second time or later it becomes a little tricky. Since we don’t want to create the data everytime we run the code we save the data in a pickle file which you can load easily. So import all the libraries and then load pickle after that code runs linearly. Same goes with model training. If you have already trained the model we save the model after training hence go to the last cell and simply load the model which already has required weights.

Images are colored and of size 128x128x3. Here 3 is the number of color channels.



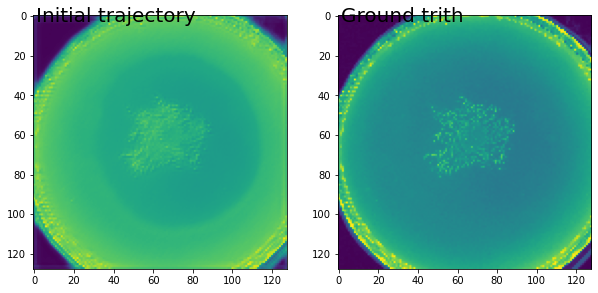
Training the model: Here since we only have 14 movies the data to train is not that high. Also we don’t really see a lot of difference between the first and the last frame because the growth of bacteria is so slow.

Speculation: We should see the images being recreated with minimal changes. This model will only be useful to predict a few frames ahead.

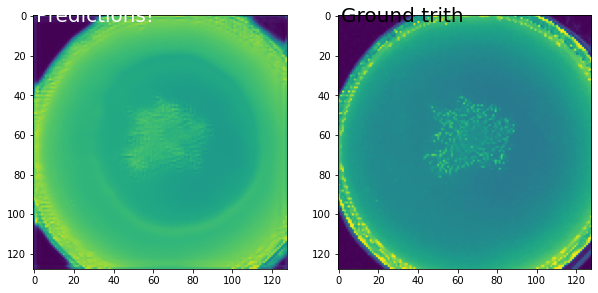
Why ConvLSTM: Because why not it eliminates creating multiple models and makes the task of prediction much easier. It just makes the programmer’s task easier.

Why NOT ConvLSTM: IT IS A BIG BOI MODEL - it takes a lot of processing power and the output is not that impressive compared to VAE+LSTM models as we will see.

Output: We see the prediction on the left and ground truth on the right. The model is able to draw the bacteria. It is difficult to tell if it is making accurate predictions as the images used to make predictions are similar to previous images the model might be just copying the input images. Since this model takes the complete image as the input the dimensionality of the input is too high to implement a controller on this.

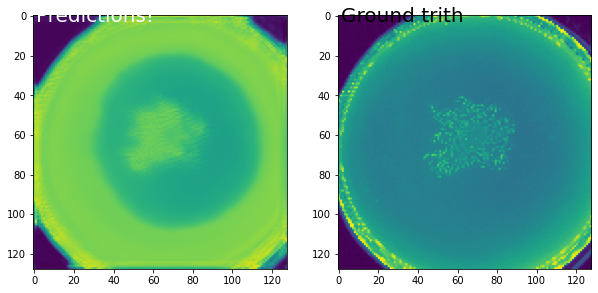


First prediction



Prediction using the previous prediction as input

We can see that the image get blurry as we try to predict more into the future



Last output from the model