Early-Stage Well Image Recognition for Cancer Identification

Sandhini Agarwal
Stanford University
sandhini@stanford.edu

Fahim Tajwar
Stanford University
tajwar93@stanford.edu

Darian Martos
Stanford University
dtmartos@stanford.edu

Abstract

With the advancement of deep learning in computer vision, the task of cancer cell identification has been aided by the use of CNNs to label cancer growth. In our project, we explore well images of cancer cells to identify cancerous growths, or other anomalies, that arise after an interval of a few days. We look into the performance of models throughout different time intervals, in order to better understand the onset of cancer during earlier-stage identification, and try to predict what the future state of the cancer cells is going to be. A common problem in medicine is understanding whether early detection of cancer, or cancer-like growths, is a good predictor for the onset of cancer in a patient. With the assistance of our mentor, PhD candidate Alexandra Sockell of Stanford's Genetics Department, we were provided a dataset of varying time intervals with labeled wells to examine this problem. We use models like logistic regression, CNNs, and mixed CNN models like CNN + LSTMs and 3D-CNNs. We compare across these models to determine what factors best aided cancer cell recognition, and then compare the different data inputs to determine if earlier-stage identification influences cancer diagnoses. Due to limits in our dataset, our accuracy was limited, but the best performing model was a CNN model trained on day 5 *images of cancer cells, with a total accuracy of* 75%*. For future works, we recommend models that capture more sequential processing of timed data such as well images through two week periods, and also gather a larger dataset. Different variations of 3D-CNNs, or CNN and RNN mixed models, would best suit this task.*

1. Introduction

Our project uses images of cancer cell wells to check for the growth of cancer in these cells after a two week period. We aim to build a classifier that can carry out early-stage cancer detection by focusing only on images within the 0-5 days time frame of the 14 day period. Within a few days, there are notable changes in the cells that can tell us the potential growth of cancer in the well, and so our project relies on this biological intuition to establish a classification baseline for these wells. Our task of early-stage cancer detection across thousands of images eliminates the need for more human identification across the wells. Thus, our task is relevant to health care and medicine, as the development of cancer cells within these wells can provide signs of cancer growth within the human body.

This project was inspired through our mentor, PhD candidate Alexandra Sockell of the Genetics Department, and work done by former students of CS 231N, as discussed in the related works section below.

Our results can be found in depth in Section 5 under our experiments. To summarize, we found that the models varied drastically in the way they performed across different labels. While total accuracy is usually an indicator of the strength of a particular model, our total accuracy is not the holistic metric we want use to portray our results, due to the lack of data in our final dataset. We included results of accuracies from other labels in order to portray the varying degrees of success that we had across the different models. Interestingly, given our limited dataset, we found that the more shallow models (like logistic regression, our first CNN model, and the shallow 3D-CNN) performed best relative to the deeper models from before.

2. Related Works

Our problem was partly influenced by the work in [1], in that we intend to work in a segmentation-based task for the well images for cancer detection. What is different from [1] is the availability of labels of our well data. The presence of labels within our dataset provides us convenience in that we do not have to augment the data as the author of [1] does in his project.

The problem of identifying growth in the cell wells is also similar to work we had found by Google, as exemplified in [11] and the related model in [5]. The model in [5] utilizes GoogLeNet to identify patches within colored images, which is similar to our intended approach. However, as described in Section 3, our dataset consists of gray images rather than colored cell images that can utilize heat maps for cancer identification. Even though our image

dataset is very different from that of CSAIL and Google, we are considering using the GoogLeNet methods such as Inception [5] as a possible classification method in the future.

One model that looks especially promising is a model proposed in [7]. This model includes four neural networks, and is done through CT scan imaging. The first is a nodule detector, which accepts a volume crop from a CT scan and identifies nodules. The second NN is known as a malignancy detector, which classifies nodules as benign (non-cancerous) or malignant (cancerous). Then, there is a nodule classifier, which accepts individual nodule volumes and similarly classifies them as benign or malignant. Finally, the last NN layer is a patient classifier, which accepts features from the malignancy detector and nodule classifier and yields the probability of a patient having cancer. While our dataset is vastly different from the one used in [7], the model is unique in that it is personalizable to a given patient and the associated problem areas in a scan.

Ultimately, the models we worked with included 3D CNNs as in [6], which are explained in depth in Section 4.4. For our CNN architectures, we worked with a variety of different models. Some of our basic models, as described in Section 4.3, are models that arise from those in the course CS 231N, Stanford University. We also train on a VGGNet model similar to that in [9], and also work with 3D-Architectures similar to those in [4]. Finally, we explore a model that intertwines CNNs with LSTMs for sequential processing, and try to combine the advantages of RNN systems similar to [3] with the advantages of convolutional neural nets. LSTMs were first introduced in 2000 by Gers, Schmidhuber in [2], and have been influential in natural language processing and computer vision since then thanks to the memory and forget mechanisms that underlie the architecture. One of the most important uses of LSTMs in computer vision has been in visual captioning, LSTMs are also helpful to analyze time series data, and we use a CNN + LSTM model to predict the final cancer cell growth type from initial images. LSTM models also are useful for NLP+Vision tasks such as image captioning in [10], and the ability to combine vision with the sequential data of language inspired us towards a model that can model time-sequential imaging data.

3. Dataset and Preprocessing

3.1. Milestone Dataset

Our dataset during our milestone consisted of images for 4800 single gastric cancer cells growing in different microwells. Each of these cells had 14 corresponding black-and-white images with each image corresponding to one day in a two-week time-frame. Thus, in total we had 67200 images in our dataset. These images were collected by performing microscopy at 10X magnification daily in a grid over the entire microwell array. The images are then stitched together, and individual microwells are partitioned out for the analysis [4].

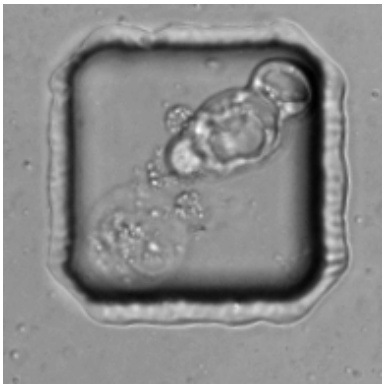

Figure 1. Grows Dense Example

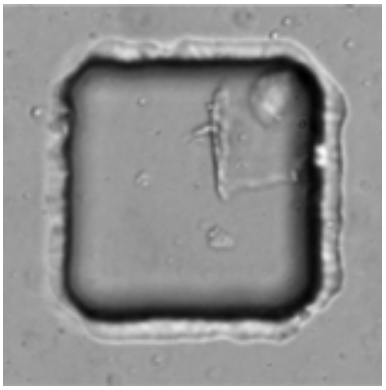

Figure 2. Debris Example

Each cell had one of 6 labels - "transient cells," "debris," "cell dies," "grows sparse," "edge artifact," and "grows dense". We are used 800 cells out of the 4800, for the purpose of our milestone, because only 800 of them were labeled. The images are in png format, but for certain models we transform them to RGB images.

3.2. Data Preprocessing

We had to carry out a series of steps for data preprocessing. Our mentor, Alexandra, provided us with labels for 800 wells in an Excel file. After that, our preprocessing consisted of two different parts. We used the "pandas" package to read and process the excel file, read the labels for different images, classify each label with a numeric value, and create an image to label map. The second and most important pre-processing was concerning the actual image data. We chose the images from first five days as our input. We used PyTorch functionalities to read these images of well 0 to well 799, day 0 to 5, and store them in appropriate torch tensors, and other PyTorch functionalities (reshaping, etc.) to finish preparing the image dataset for a

learning model to train on.

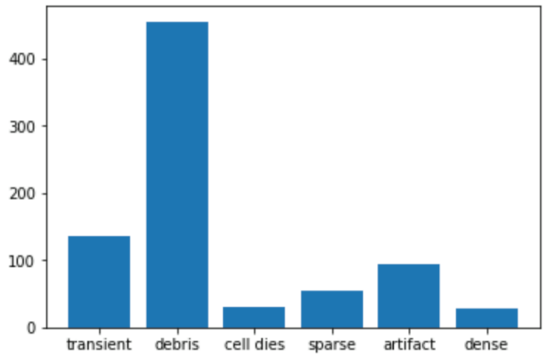

Figure 3. Unanticipated Class Imbalances in Milestone Data

3.3. Final Dataset

Ultimately, to achieve better results with our models, we had to scrap about 450 labeled wells that turned out to be debris, in order to analyze the 350 labeled image wells that had contained real cancerous cells. Because we scrapped debris cells, and other anomalous well images such as transient cells, the labels that we train over are limited to "cell dies," "grows sparse," and "grows dense."

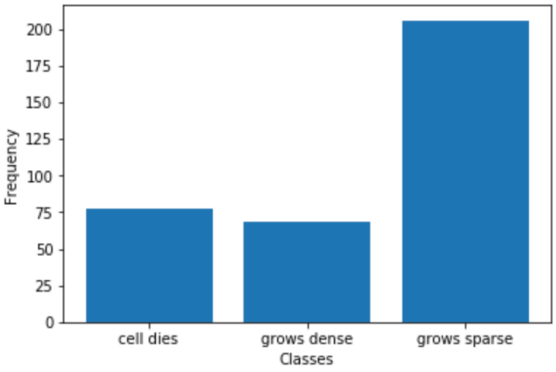

Figure 4. Class Distribution in Final Project Data

Other than this change in our labeling and the minimization of our dataset, the dataset remains the same and was collected through the same procedures as it had been during the milestone. The new dataset now has the following distribution of labels: exact numbers: cell dies: 77, grows dense: 68, grows sparse: 206.

3.4. Data Augmentation

Since our dataset is so small, we decided to use data augmentation on it. We used an indirect data augmentation method, where we keep our original dataset, but during training, each image goes through a series of transformations, which include random flips, rotations, cropping, normalization before feeding it into the trainer. This provided more variation in our dataset. After experimenting with different transformations, the best results were obtained from a transformation composed of random horizontal flips, vertical flips, and normalization.

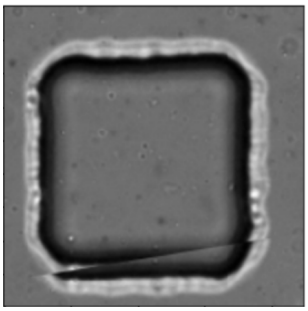

Figure 5. Image before transformation

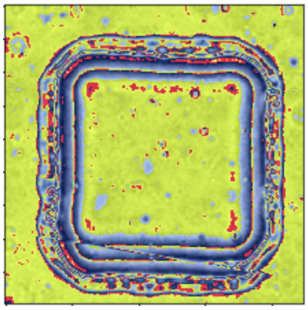

Figure 6. Image after transformation

3.5. Dataset Split

We put 80% of our data in training, 10% in validation, and 10% in test set. We do not use cross validation, since there is not enough data to having meaningful data folds.

4. Methods and Approaches

4.1. General Approach

We classify cells into three classification categories based on the type of growth in the wells: "cell dies," "grows sparse," and "grows dense". We used only images from Day 1 - Day 5, because the wells up until Day 5 reflect the early stages of cancer cell growth already, and is already an influential sign of cancer detection.

Thus, our inputs for our model are image data (either a single image, 5 images depending on which experiment we are carrying out) and our outputs are labels corresponding to how the cell would grow. We carry out a series of experiments and compare results by building models that take in images only from Day 1 - Day 5, and images only from Day 5. This separation between the data helps us discern how much predictive power there is in the early stages of cell growth, an important problem in healthcare today.

4.2. Loss Function

For all models, we chose to use the weighted cross entropy loss function. Cross entropy loss is a very good loss function and can be widely used in multi-class classification, and we chose to use the weighted version since our classes are highly imbalanced.

$$loss(x, class) = w[class](-x[class]+log(\sum_j exp(x[j])))$$

We experimented with various weight functions, and finally chose to use the following weight function: let f_i be the frequency of data points for class i, then the weight for class i becomes:

$$w_i = \frac{\max_j f_j}{f_i}$$

This way, we penalize the misclassifications for less populous classes more, thereby prevent making a totally biased predictor.

4.3. Logistic Regression Baseline

As a baseline method, we built a logistic regression model with stochastic gradient descent. We built two variations of this model - the first model took in only one image from Day 5 as its input. The second model took images from Day 1 - Day 5 as input. We chose this as a reasonable baseline given the small size of our dataset.

As a baseline method, when we were working with time-series images from Day 1 - Day 5, we simply concatenated the 5 images together before feeding it into the logistic regression model. This turned our Tensor into a 4-D tensor which we had an extra dimension for the stacked images.

4.4. Base CNN Models

As a preliminary model for our milestone, we built a CNN which took in Day 5 image data with the following architecture: (Conv $\rightarrow$ Batch Normalization $\rightarrow$ ReLU $\rightarrow$ MaxPool)*2. This model achieved 53 percent accuracy on the dataset. ReLU works as our non-linearity in the neural network, and "Conv" represents a convolutional layer.

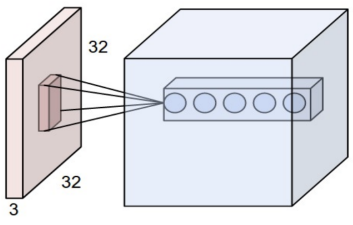

Figure 7. A convolutional layer [8]

As we developed our models further, we incorporated CNN models from Assignment 2 into our project to examine the effects of simple CNNs without 3D-convolutions or mixed models with RNNs. One basic CNN model, known as CNN 1 in the below Tables 2-5 in Section 6, is a CNN that includes the following layers: (Conv $\rightarrow$ ReLU $\rightarrow$ MaxPool)*3 $\rightarrow$ (Affine) $\rightarrow$ (Weighted Cross Entropic Loss). The other basic CNN, known as CNN 2, is similar to CNN 1 except we run through a deeper layer of (Conv $\rightarrow$ ReLU $\rightarrow$ Conv $\rightarrow$ ReLU $\rightarrow$ MaxPool)*3 before running through the affine and weighted cross entropy layers.

4.5. VGG

The VGG network, as described in [9], is a very deep convolutional network that allows for greater visual representations in computer vision tasks. The layers of a VGG, as stated in [9], are: "A stack of convolutional layers, ... followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer." Here we change the number of classes to 3, and train the model from scratch. With deeper layers, we should be able to generalize to features outside those of simple CNNs, by learning more broad representations of the well images.

4.6. CNN and LSTM Network

We tried a CNN followed by an LSTM network. This is because we hoped that the CNN would help us with feature extraction across the images and the LSTM would help support sequence prediction for the time-series data. We passed each individual image into the CNN distributed across time to compute the features of each image and then feed these feature vectors into the LSTM as input.

Our CNN architecture consisted of a a conv layer followed by Relu followed by maxpool. This was repeated three times. The features extracted from this were then fed into a 5 layer LSTM.

4.7. 3D-CNN

With a 3D-CNN, we use 3D-Kernels in CNNs, which are common with video data. With our dataset, since we have a variety of 2D-images applied in sequence, a 3D-CNN would be useful in mapping out the timed progression of the wells and various growths that occur during the early-stage period. One potential model that we have looked into for our work includes 3D-SqueezeNet [4], one of the first resource efficient 3D-CNNs that includes a few max pool layers and "Fire" blocks, which are stacks of CNN layers concatenated and then put through a ReLU nonlinearity. We explore some simple 3D-CNN architectures. 3D CNNs are widely used in video analysis and video classification. We realized our data is a video of the wells, with frame rate

of 1 frame per day. So we stacked the image of a particular well in different days. The stacked pytorch tensor had shape $C \times T \times W \times H$, where T is the number of days, C is the number of channels, W and H are the height and width. For our dataset, a particular well would generate a torch tensor of shape $(3, 5, 224, 224)$. We train and test 3D CNNs on these tensors. Note that, we did not do any data augmentation for the dataset used by 3D-CNN, which can explain the relatively poor performance of this model as we will see later.

5. Experiments

5.1. Milestone Experiments

The three models we tested - logistic regression with Day 5 image, logistic regression with concatenated Day 1 - Day 5 images, and CNN with Day 5 all achieved accuracy between 50-60 percent. However, when we began probing into the results of our models, we realized that our dataset was heavily skewed, which we had not anticipated. Thus, despite our high accuracy, it is unclear whether our model is learning anything. Our dataset during the milestone had the following data balance: transient cells: 136, debris: 455, cell die: 31, grows sparse: 55, edge artifact: 94 and grows dense: 29. To aid in this problem, we ultimately removed debris, transient cell, and edge artifact results that heavily skewed the data away from well images without actual cells.

5.2. Final Experiments

As indicated in Section 3.3, our dataset was drastically changed in order to more carefully focus on image recognition for images with actual cells in them, and not unusual well images such as those with debris or edge artifacts. Thus, we removed three labels from our dataset and focused on classifying cells only into 'grows sparse', 'cell dies' and 'grows dense' categories. Our dataset now consisted of 351 samples of 206 grows sparse labels, 77 cell dies labels and 68 grows dense labels.

We carried out all the experiments using the augmented dataset with the exception of the 3D CNN and the CNN followed by LSTM model. This is because the time series added a 5th dimension to our data structure and did not allow image augmentation.

Additionally, we carried out the 3D CNN and CNN+LSTM experiments with Day1 to Day5 images. All other experiments were tuned and carried out with Day 5 images because we discovered that the models performed better on single images as opposed to stacked time sequenced images.

5.3. Results from Experiments

The results are on the tables listed. We chose to report the total accuracy and per class accuracies, since our dataset is highly imbalanced, total accuracy is not entirely meaningful. We did not report a confusion matrix, because our test set has only 35 images and 3 classes, so a confusion matrix would not provide additional information. The results tabulated are obtained from the best models trained on training set, fitted on validation set, and finally tested on our testing set.

Method	Total Accuracy (%)
Logistic Regression	52
CNN 1	75
CNN 2	38
VGG	25
CNN + LSTM	36
3D CNN	54

Table 1. Final Project Total Accuracy

Method	"cell dies" Accuracy (%)
Logistic Regression	80
CNN 1	83
CNN 2	50
VGG	0
CNN + LSTM	8
3D CNN	77

Table 2. Final Project "cell dies" Accuracy

Method	"grows sparse" Accuracy (%)
Logistic Regression	31
CNN 1	68
CNN 2	33
VGG	25
CNN + LSTM	0
3D CNN	52

Table 3. Final Project "grows sparse" Accuracy

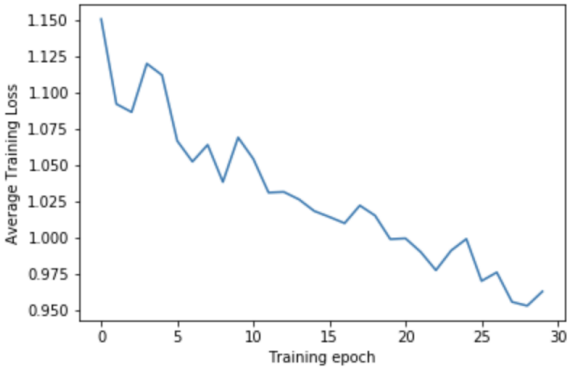

Figure 8. Loss Graph For CNN 1

Our best model turns out to be the CNN 1, which is not that deep. The reason is, we hypothesize, the lack of data. There is not enough data to train the deeper, more complex networks, which easily results in overfitting and bad results.

Method	”grows dense” Accuracy (%)
Logistic Regression	44
CNN 1	75
CNN 2	33
VGG	50
CNN + LSTM	100
3D CNN	33

Table 4. Final Project ”grows dense” Accuracy

5.4. Hyperparameter choice

The only hyperparameter we tuned in our models is the learning rate. We used all of our images for the gradient descent algorithm, and did not use any advanced update rule (like Adams or Adagrad), instead we used simple gradient descent algorithm. So batch size was equal to total number of training points and was not a hyperparameter. We chose the learning rate by training different models using different learning rates, testing them on the validation set, and keeping the best model. Then we only use this best model on our test set and report the results from it. We do not do cross validation since the validation set is really small to have any meaningful data folds.

5.5. Error Analysis: Visualizing misclassification

We decided to visualize a few of the misclassifications from the CNN 1 model. It turned out that, we were having the highest number of errors for the grows sparse class, our most populous class, and most of the data there were being classified as grows dense.

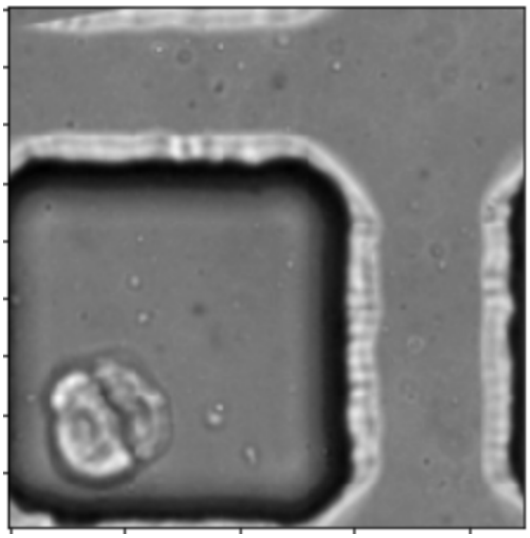

Figure 9. Misclassification 1: Sparse is classified as Dense

We think the reason is that these two classes are harder to predict, i.e. it is hard to say which class a cancer cell will turn out to be in later course of its developments between

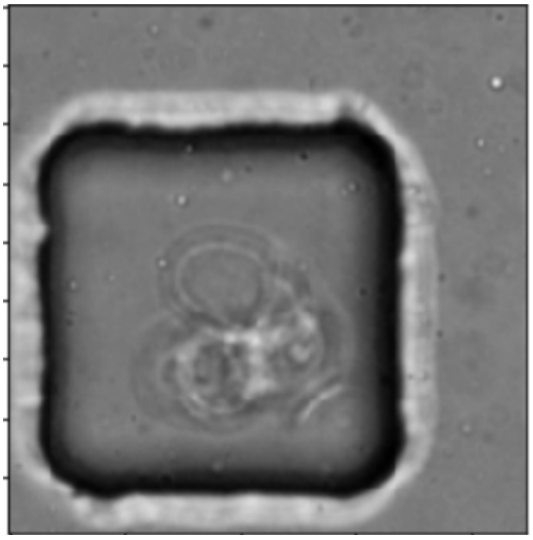

Figure 10. Misclassification 2: Sparse is classified as Dense

these two classes, since in earlier stage they look pretty similar. One quick test was to classify them both into a same group, and run a binary classification on two classes, ”grows sparse” and ”cell death”. We hypothesized that this will give us a very good result, and it turned out to be correct. For this binary classification task, the overall accuracy jumps up to approximately 80%, with per class accuracy increasing up to approximately 70%.

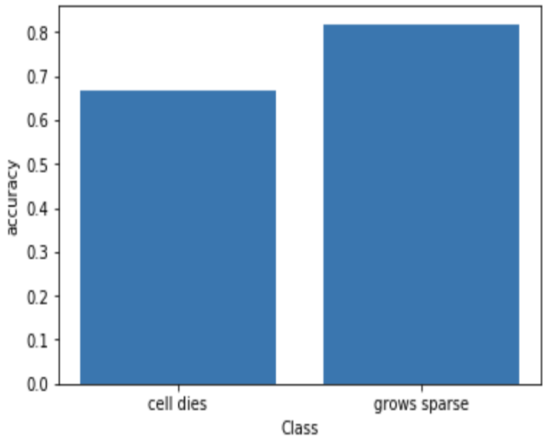

Figure 11. Binary classification results

Another thing we observe is that cell growths often change trajectory in between the initial time frame (day 1 to 5), and final time frame (day 14). Most often, many dense growths disintegrate into sparse structure, as we see in the examples below. Here, on day 5 the growth is very dense, but on day 13 it disintegrated into a sparse structure. Looking at the misclassifications, we see that this is often the reason for examples where our model behaves poorly.

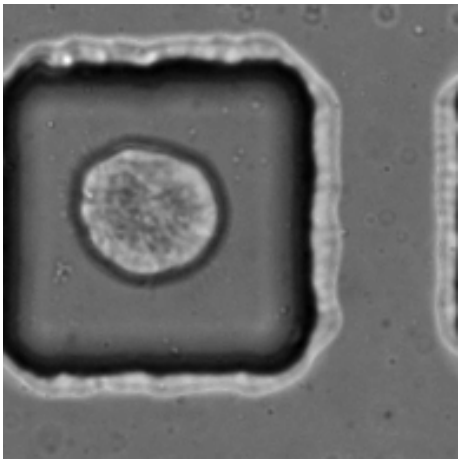

Figure 12. Well 1037 day 5: cell growth forms dense structure

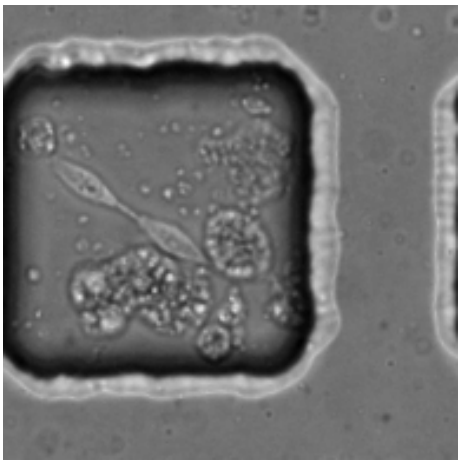

Figure 13. Well 1037 day 13: Disintegration into sparse structure

6. Future Work

We hypothesize that this sudden growth trajectory change described in 5.5 might be predictable from early stage images. Looking at some of the cell growths where this change occurs, we noticed that mostly cells that quickly grow into a large dense structure disintegrates later, however cell growths that are smaller in shape but dense nonetheless, do not generally disintegrate. We aim to have a larger dataset so that this variation, if it actually exists, is better represented in our dataset. We also assume that more complex architectures with more data might be able to better learn this features, so we plan to experiment with them in order to have better results.

7. Conclusion

From our milestone results, we found that a simple logistic regression on just Day 5 cells was able to label the wells most accurately.

Due to the limits in our dataset, our accuracy appears lower than it could be since we were only confined to 250 workable examples. In the future, the most obvious workaround to achieve a higher accuracy in cancer cell recognition is to ensure there is enough labeled data to work with.

However, we were able to achieve 75 percent accuracy with our CNN1 model. This model performed the best despite being one of the simpler models. Given the size of our dataset, the more complex models were unable to perform well and gave highly skewed accuracy. For example, the CNN+LSTM model gave 100 percent accuracy for 'grows sparse' but only 8 percent and 0 percent accuracy for 'cell dies' and 'grows dense'. Thus, we found that simpler architecture designs can sometimes be the most powerful depending on the nature of the task and dataset, especially in smaller datasets that lead to less feature representations.

References

[1] A. Abdulhamid. Semantic segmentation with histological image data: Cancer cell vs. stroma, 2017.

[2] F. Gers and J. Schmidhuber. Recurrent nets that time and count. 2000.

[3] K. Gregor. Draw: A recurrent neural network for image generation. 2015. Available at https://arxiv.org/pdf/1502.04623.pdf.

[4] F. Iandola. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. 2017. Available at https://arxiv.org/pdf/1602.07360v4.pdf.

[5] A. Khosla. Deep learning for identifying metastatic breast cancer. 2016. Available at https://arxiv.org/pdf/1606.05718.pdf.

[6] O. Kopuklu, A. Gunduz, and G. Rigoll. Resource efficient 3d convolutional neural networks. 2019. Available at https://arxiv.org/pdf/1904.02422.pdf.

[7] K. Kuan and M. Ravaut. Deep learning for lung cancer detection: Tackling the kaggle data science bowl 2017 challenge. pages 2–8, 2017. Available at https://arxiv.org/pdf/1705.09435.pdf.

[8] F.-F. Li, J. Johnson, and S. Yeung. Lecture slides, cs231n, stanford university, 2019.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2015. Available at https://arxiv.org/pdf/1409.1556.pdf.

[10] J. Song. Hierarchical lstms with adaptive attention for visual captioning. 2018. Available at https://arxiv.org/pdf/1812.11004.pdf.

[11] M. Stumpe and L. Peng. Assisting pathologists in detecting cancer with deep learning. *Google AI Blog*, 2017.