Lung Cancer Detection Using CT Scans

Jacob Barney

Introduction

Computer Vision techniques have played a part in many recent medical advancements. Some notable examples are skin cancer detection from your phone, Covid-19 detection using lung x-rays, and tumor detection. The example that we will focus on is tumor detection, specifically tumors in CT scans of lungs.

Background and related work

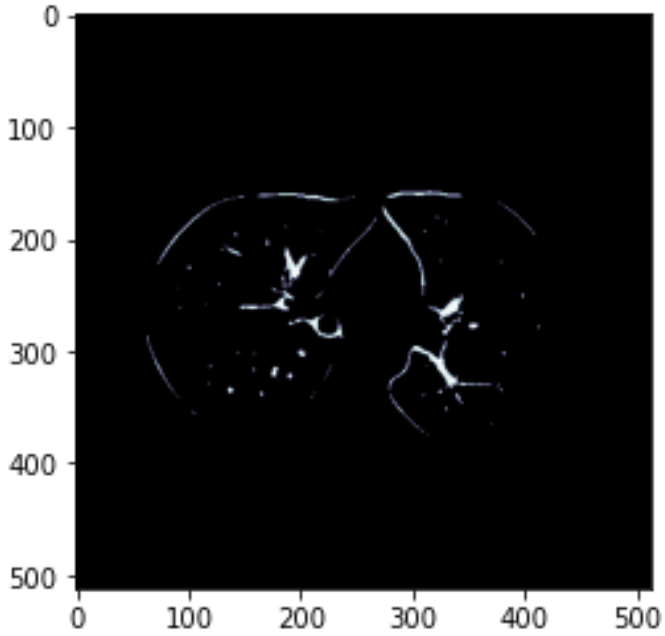
There are many published techniques for identifying tumors and indicating whether they are malignant or benign. I attempted to reproduce a process published by Alakwaa et al. that uses a U-Net convolutional neural network (CNN) for identifying nodules in stacked 2D cross-sections of CT scans (like the image to the right) and then uses a simpler CNN for predicting the 3D stacked nodules might be malignant or benign. They end up with an accuracy of 86.6%, an 11.9% false positive rate, 14.7% false negatives and they claim “almost all patients are classified correctly” (Alakwaa et al). The data that they use for a 3D scan with malignant or benign labels was part of the Data Science Bowl by Kaggle, however the data used is no longer available “due to data set usage restrictions” (“Data Science Bowl 2017”). Due to the missing data I will instead be using 2D scans from Kaggle’s IQ-OTH/NCCD - Lung Cancer Dataset. My new goal is to determine how using the 2D images compares to using 3D for training the cancer vs no cancer model. If it can perform as well then it may be a more efficient algorithm to use.

Many other interesting studies have been performed on this topic. One very interesting study is by Ritchie et al. where they wanted to see if they could mark nodules as cancerous or non-cancerous using computer vision before a technician sees it and see if they could increase the speed and/or accuracy with this new workflow. Another study, which I wanted to implement as an addition to my chosen study, uses a texture appearance model to better identify nodules in the CT scan image. The paper by Shariaty et al. did not give much information into their process so I wasn’t able to replicate it.

Methods

The first step in the process is finding potential nodules on CT scans. Pre-processing the 3D image data is difficult but very important. The images come in MHD files along with corresponding RAW files that have additional information.

After importing I blurred the image with a gaussian blur with sigma of (5,5). The images are in hounsfield units (used for this medical imaging) so to keep the tissue and bronchioles of the lungs I used a threshold on the image of -900 to -320. The study doesn't give exact numbers but these worked well in this case. I removed any remaining tissue to find potential nodules using the clear_border segmentation and a Roberts filter with binary_fill_holes. The study doesn't say what it uses, but these worked well. I got the ideas from the Candidate Generation and LUNA16 preprocessing on Kaggle. We then use this as a mask to get the original image with only the remaining nodule candidates. We then set the image values between 0 and 1, use 3d spline interpolation to resize the image to # of images x 256 x 256 to get it into the right input format for the neural network, and use zero means to make the data symmetric. The image to the right shows potential nodules as found by this method.



The output data for training is a binary array of the same size as the input. The coordinates and diameter of each output nodule comes from the annotations.csv file included with the data, but you all need the origin and spacing which can be extracted from each image and is unique to each. You will need to use an equation to get the coordinates and diameter for your data. The coordinates are the absolute value of the coordinates from the CSV file minus the retrieved origin, divided by the spacing. The diameter is the absolute value of the diameter divided by the spacing. A 3D box of 1's is set in the location and size of each nodule, while the rest of the array is set to 0.

The U-Net CNN is very memory and computationally expensive so I elected to only pass in 2D slices of the 3D arrays that contain any nodules as training data. My computer with 16GB of memory could not handle training the model. Attempted to speed it up with my graphics card plus CPU memory and the memory problem still persisted so I had to train the model on Google's Vertex AI. The U-Net architecture used in the study can be found in Alakwaa et al, however I used one that is slightly more complicated that is close to the one used in a video by DigitalSreeni. The model was trained with 99.9% accuracy and it has the following structure:

- Input layer of 256 x 256 x 1
- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation
- Dropout layer of 0.2
- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation

- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2
- 2D convolution layer with 128 filters and a kernel size of 3 with relu activation
- Dropout layer of 0.2
- 2D convolution layer with 128 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2
- 2D convolution layer with 256 filters and a kernel size of 3 with relu activation
- Dropout layer of 0.2
- 2D convolution layer with 256 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2
- 2D convolution layer with 512 filters and a kernel size of 3 with relu activation
- Dropout layer of 0.2
- 2D convolution layer with 512 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2

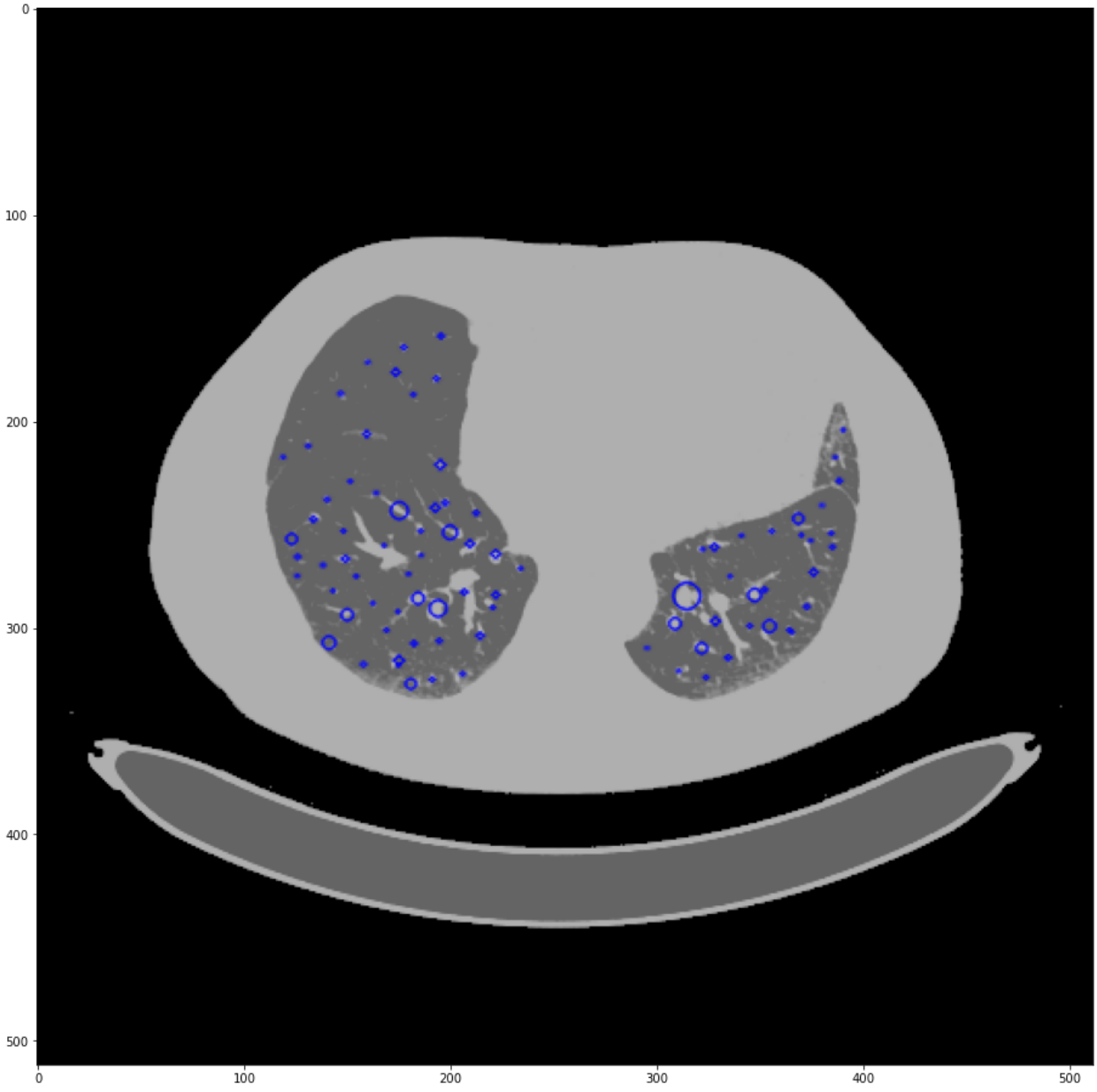
This section is call the bridge - it bridges the sides of the “U”:

- 2D convolution layer with 1024 filters and a kernel size of 3 with relu activation
- Dropout layer of 0.2
- 2D convolution layer with 1024 filters and a kernel size of 3 with relu activation

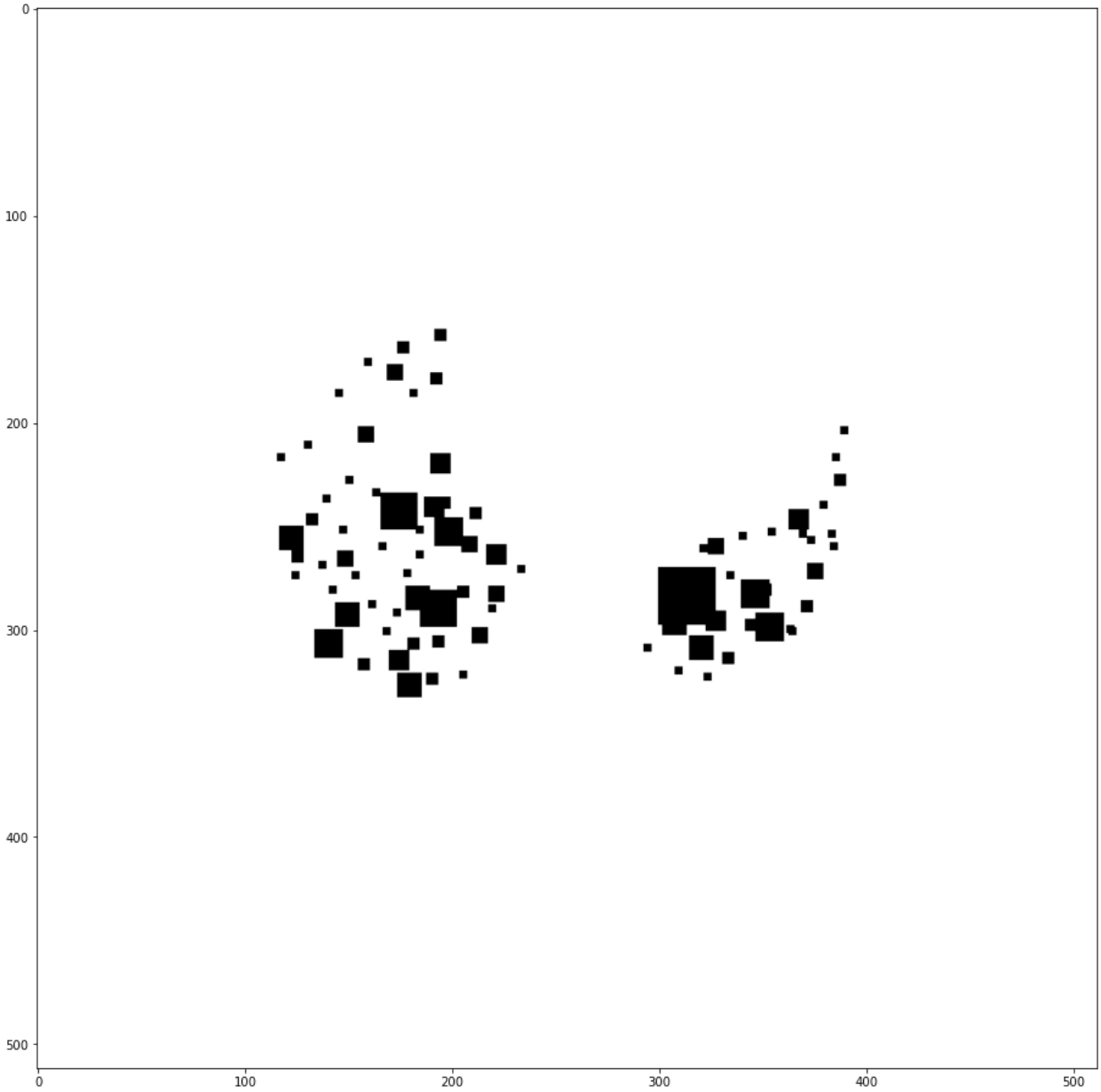
The rest is the structure is the pre-bridge bullet points in reverse, however:

- The last of each set of 2D convolutions will be 2D upsampled with a size of (2,2) with bilinear interpolation
- No max pooling layers
- Each set of 2D convolutions will start with another 2D convolution layer with the same number of filters but a kernel size of 2. The output of this layer is then concatenated with the output of the previous set of convolutions having the same number of filters on the other side of the “U”
- There is a final 2D convolution layer with 1 filters and a kernel size of 3 with relu activation
- The output layer is a 2D convolution layer with 1 filter and a kernel size of 1 with sigmoid activation to give to work with our binary output so that we can get a probability that each pixel is part of a nodule

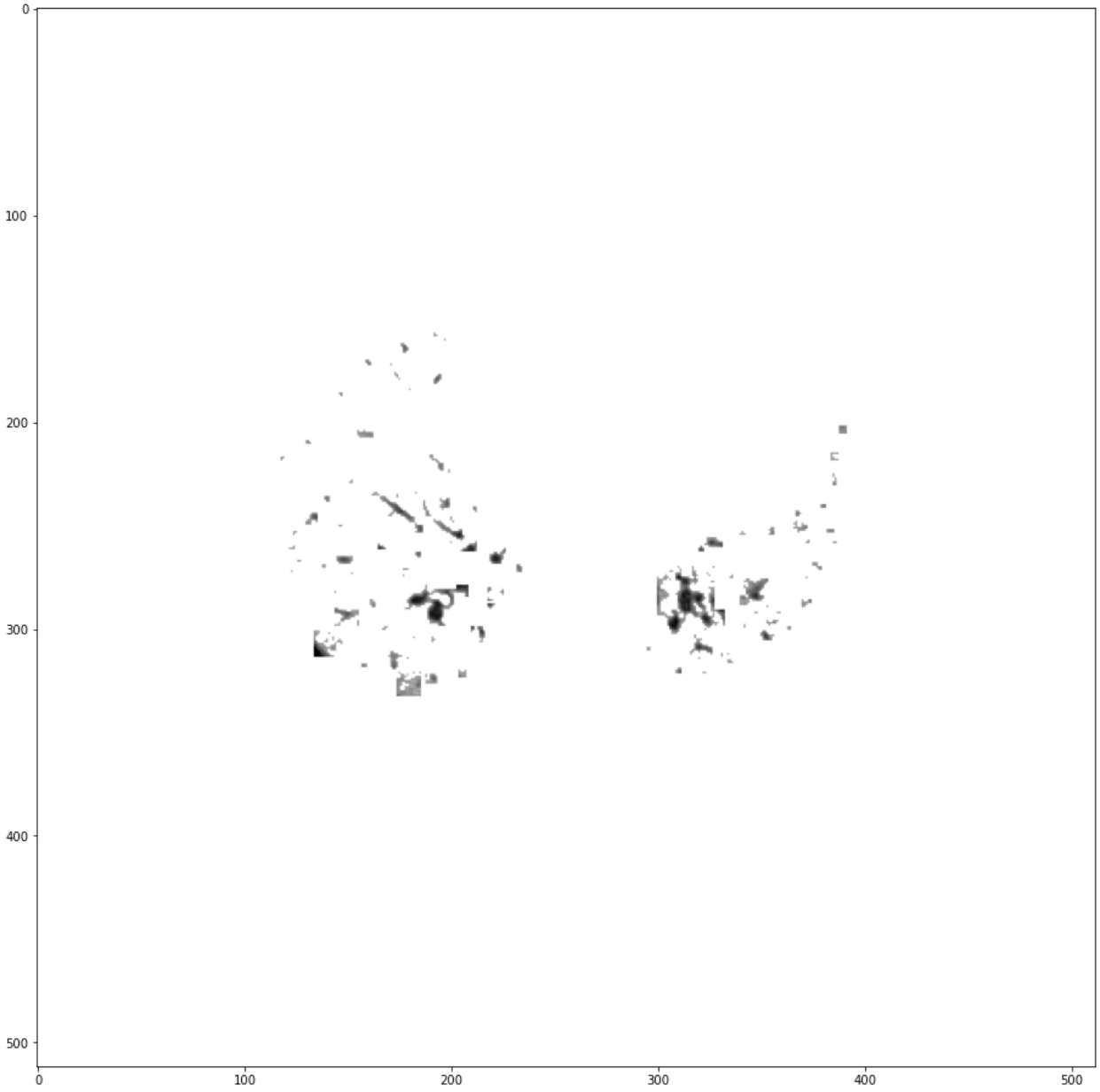
After training the model we need to apply it to the data in the 2D data set with PNG files. This data set must undergo a different preprocessing method as the images are in a different format. This time we still get the image, convert it to grayscale and use thresholding, this time a min-max of 100 to 175. Then run the image through the clear_border function again. The next step is to find any potential blobs that could be nodules that we want to keep. We use the SimpleBlobDetector function from the CV2 library adn set the parameters to not filter by inertia or convexity, but to filter by color with the color set to 255 and area filter with a max area of 1000 and no minimum. This works well to find the potential blobs, as shown in the image below:



We can take the blob coordinates and their diameters to then draw up another mask. The mask is made using a 2D array of zeros and filling in a square of 1's at each coordinate with a length and width of the diameter and get a result like the following:

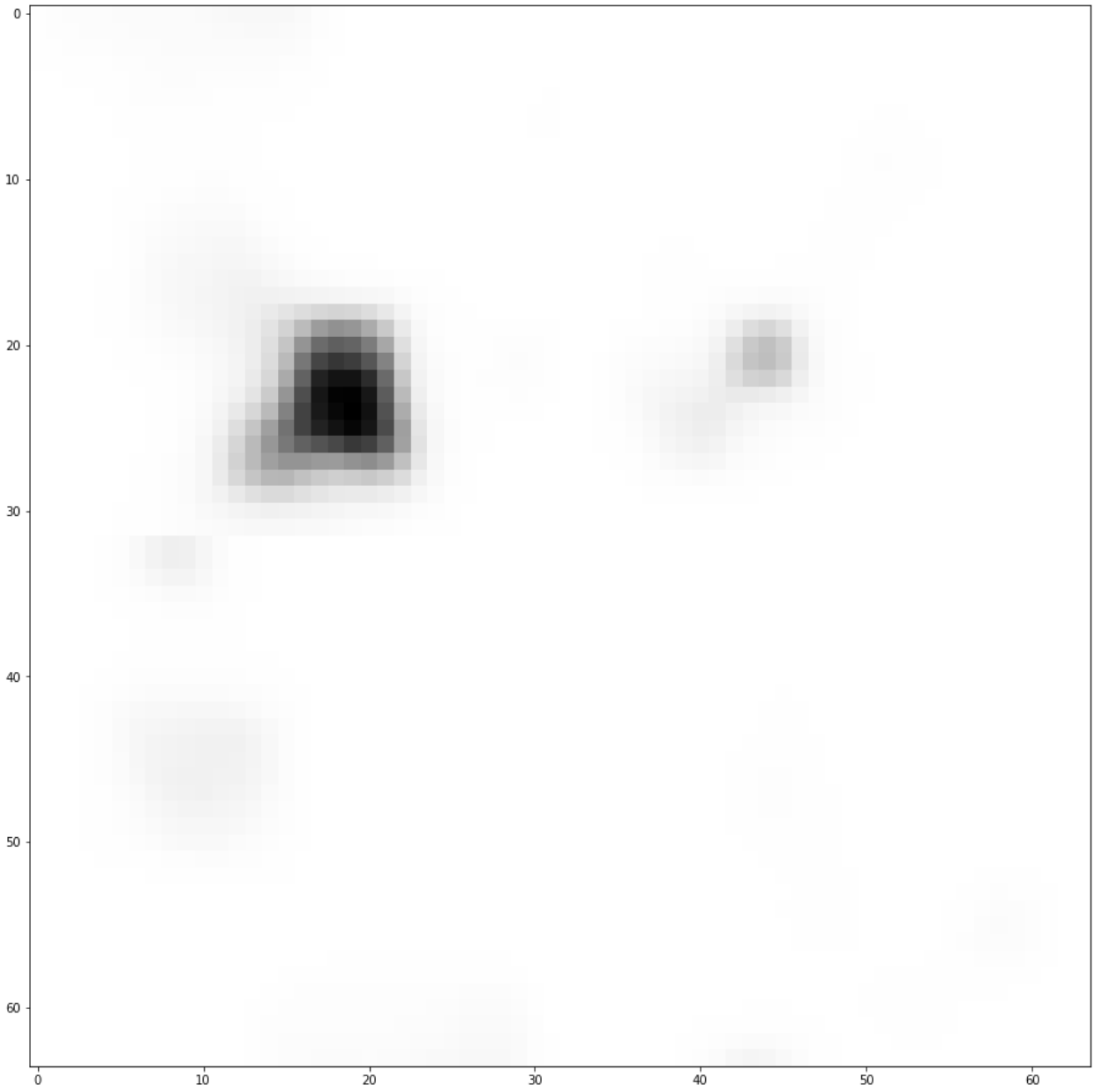


We then apply this mask to the ct scan and normalize with the same methods as before and get the final image before checking for nodules:



We run the result through the model to get a 2D array with the probability of there being a nodule at each pixel.

We use a 32 x 32 window to find the top 4 nodule candidates using the local max of the sum of probabilities, ensuring that no selected windows overlap. Each candidate image is concatenated to make a 64 x 64 array which will be used to train the cancer vs no-cancer model. An example can be found below:

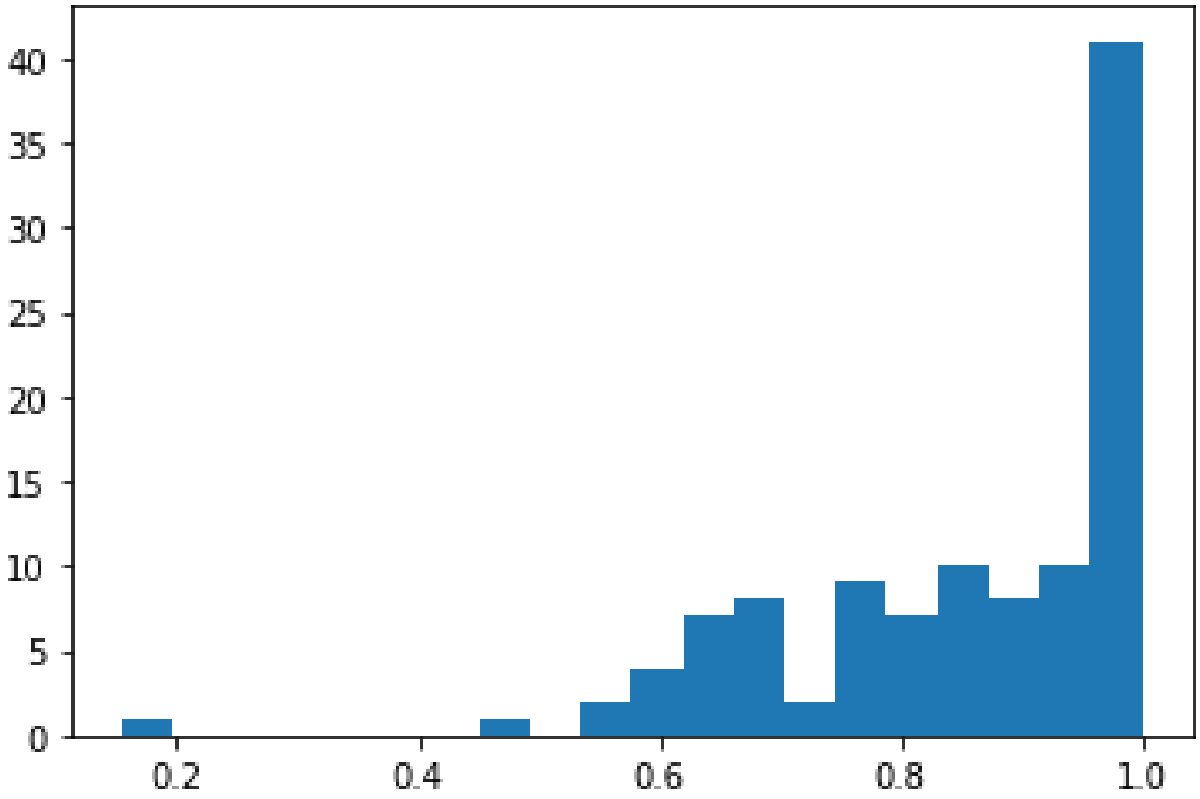


We use a basic CNN to determine the malignancy from these inputs. The neural network's architecture is as follows:

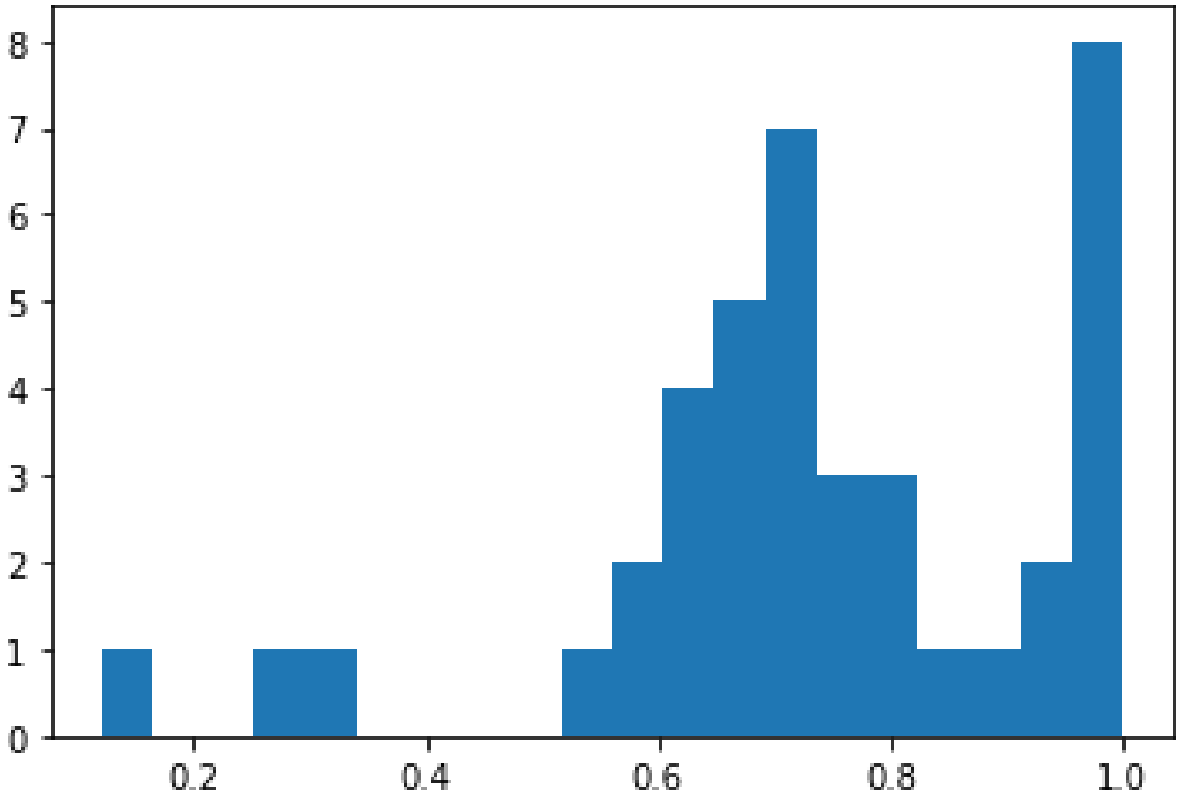
- Input layer of 64 x 64 x 1
- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation
- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2
- Dropout layer of 0.2
- 2D convolution layer with 128 filters and a kernel size of 3 with relu activation
- Max pool with a pool size of 2 x 2
- Dropout layer of 0.2
- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation

- 2D convolution layer with 64 filters and a kernel size of 3 with relu activation
- Flatten layer
- Dense output layer of size 1 with sigmoid activation

After training the data we run a test on a segment of the data that was not used for training. 110 malignant cases and 40 benign. Due to the sigmoid activation for the output layer we have an output of the chance that a nodule is cancerous or not. The malignant cases were distributed as according to this histogram:



The benign cases are distributed as seen below:



We use 80% as the cutoff, however that can be more or less, which would lead to an increase or decrease in false positives and false negatives.

Results

The segmentation portion performed well with 99.9% accuracy on 10% training data, however there may still have been some overfitting with there being multiple similar inputs due to splitting up the 3D images. The segmentation couldn't be applied to the second data set without segmenting out everything but the candidate nodes, once those were segmented out it was able to identify potential nodules.

The malignancy portion performed better than guessing, but still not as great as the followed study. The results are as follows:

- Accuracy: 0.67
- Precision: 0.84
- Recall: 0.68
- F1: 0.75
- False Negative: 0.32
- False Positive: 0.35

The biggest concern is the false negative rate at 32%. This is a life-or-death decision so 32%, especially when better alternatives exist is not good at all. The rate is about twice that which was obtained from the other study, with the false positive rate being about 3 times more. The accuracy from the other study was 86% which is much higher than ours, however it would be nice if they provided the recall so that we could compare F1 scores as these categories were lopsided in weight.

Discussion

The results partially answer my initial question which was whether I could get a result as accurate with just 2D images. The results strongly show that the results with 3D images performed better, however it is possible that they had more training data available than I did. I also had to use PNG files which means that the original CT scans had been manipulated in some way already as they come in other formats from the machines. I do think that the malignancy model could be improved without new data, but not to the same level as the other study. The way the 3D model works is essentially a boosting algorithm as each 2D layer votes on malignancy, so it makes sense that it would perform better.

Conclusion

Of all types of cancer, lung cancer kills the most people in the United States, with 131,880 estimated deaths in 2021 (“Common Cancer Types - NCI”). This research can be helpful in the quest for quicker and more accurate testing for lung cancer by showing what might and might not work. By providing a faster way to test it is possible to detect the cancer more quickly, leading to a lower risk of death. It is likely, however, that earlier detection could be more difficult as the model is trained on lung cancer nodules that can be detected and labeled by doctors. There have been pre-labeling methods tested to include both a computer-assisted and doctor-assisted element to the decision making that seems to be successful. Hopefully with the increase of data and computing power we will continue to increase the quality of care that can be provided to patients as more advancements are made with computer vision.

Data

Nodule detection: https://luna16.grand-challenge.org/Data (111 GB)
Cancer detection:
https://www.kaggle.com/datasets/adityamahimkar/iqothnccd-lung-cancer-dataset (200 MB)

Works Cited

Alakwaa, Wafaa, et al. “Lung Cancer Detection and Classification with 3D Convolutional Neural Network (3D-CNN).” *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. 8(8), 2017, http://dx.doi.org/10.14569/IJACSA.2017.080853.

“Candidate Generation and LUNA16 preprocessing.” *Kaggle*, https://www.kaggle.com/code/arnavkj95/candidate-generation-and-luna16-preprocessing/notebook.

“Common Cancer Types - NCI.” *National Cancer Institute*, 22 April 2021, https://www.cancer.gov/types/common-cancers. Accessed 5 May 2022.

“Data Science Bowl 2017.” *Kaggle*, https://www.kaggle.com/competitions/data-science-bowl-2017/data.

DigitalSreeni. *219 - Understanding U-Net architecture and building it from scratch*. 27 May 2021. *YouTube*, https://www.youtube.com/watch?v=GAYJ81M58y8.

“IQ-OTH/NCCD - Lung Cancer Dataset.” *Kaggle*, https://www.kaggle.com/datasets/adityamahimkar/iqothnccd-lung-cancer-dataset.

Ritchie, Alexander J., et al. “Computer Vision Tool and Technician as First Reader of Lung Cancer Screening CT Scans.” *Journal of Thoracic Oncology*, vol. 11, no. 5, 2016, pp. 709-717, https://doi.org/10.1016/j.jtho.2016.01.021.

Shariaty, Faridoddin, et al. “Texture appearance model, a new model-based segmentation paradigm, application on the segmentation of lung nodule in the CT scan of the chest.” *Comput Biol Med.*, vol. 140, no. 105086, 2021, doi: 10.1016/j.compbiomed.2021.105086. Epub ahead of print. PMID: 34861641.