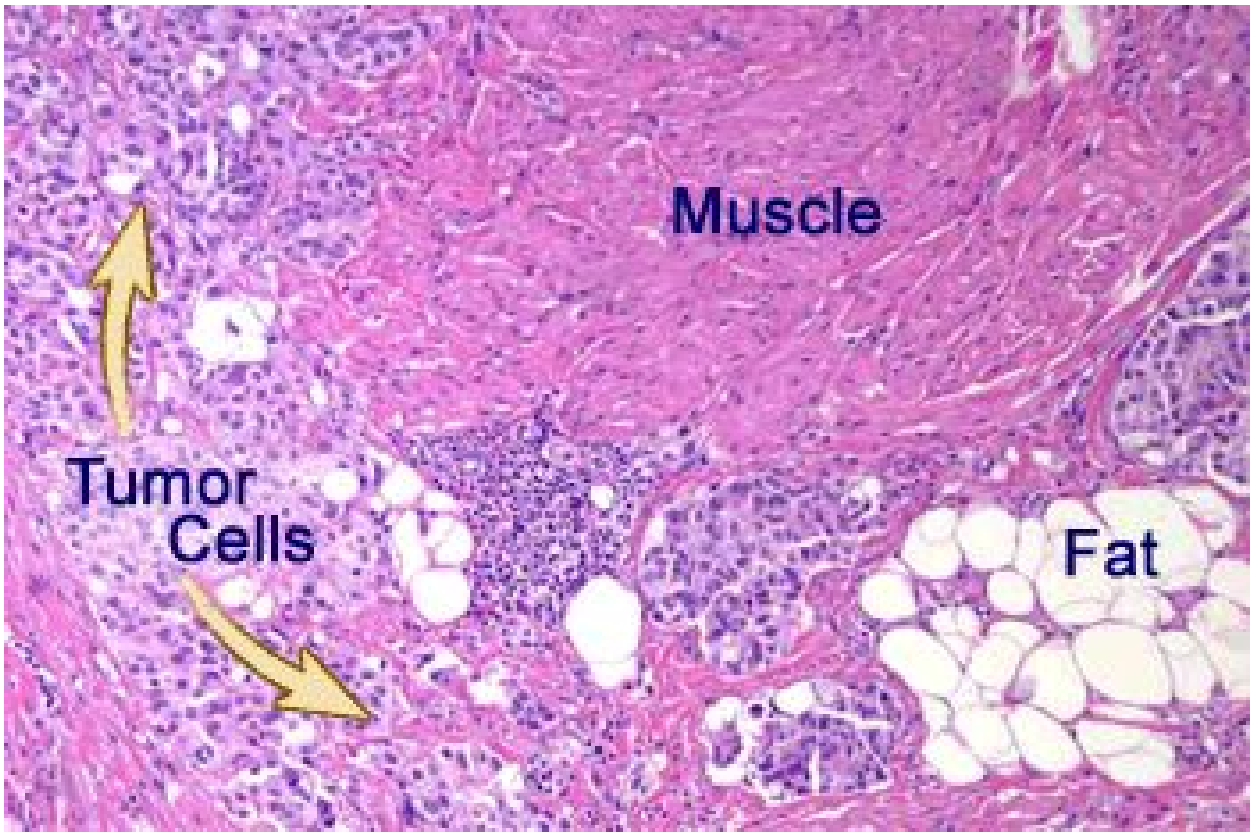
RESEARCH INTERNSHIP PROJECT

TRANSFER LEARNING IN ADENOCARCINOMA DETECTION

Naren Sridharan



Introduction

The science of Medical Image Analysis (MIA) has seen an immense change of a positive nature since the practice of transfer learning came into play. Use of Convolutional Neural Networks (CNN) in the MIA domain can be traced back to 1990s but transfer learning was introduced only after a decade into the deep learning field, and only as late as 2016 in the MIA. Reasons for this delay include lack of computational power, lack of a benchmark transferable dataset like ImageNet (2012), and mainly due to the doubts of transferability of

features used to detect large objects into the MIA field that deals with microscopic objects. This project addresses the latter concern by taking a specific sub-domain in the MIA, adenocarcinoma **detection** from histopathological images, for three different organs and uses transfer learning based both on ImageNet and Adenocaricnoma data itself to find whether there are any significant differences between the two methods.

The main objectives of the project include,

1. Creating benchmark models trained from pre-trained ImageNet weights for all 3 datasets
2. Create transfer learning models from the pre-trained models mentioned in objective 1 and fine-tuned on other cross-organ datasets.
3. Visualize differences between benchmark models and fine-tuned models to obtain insights.

Adenocarcinoma

Adenocarcinoma is a type of cancer that forms in the glands, the cells that secrete substances within or out of the body.

TYPES

Adenocarcinomas begin in glands but can spread to other areas of the body. Glands secrete various fluids into tissues that line many organs of the body. Adenocarcinomas account for the majority of cancers in the following areas:

- breast
- lung
- prostate
- pancreas
- colon

A rare type of adenocarcinoma called adenoid cystic carcinoma begins in glands in the head, such as the sinus glands. It is slow-growing cancer but can spread to the skull.

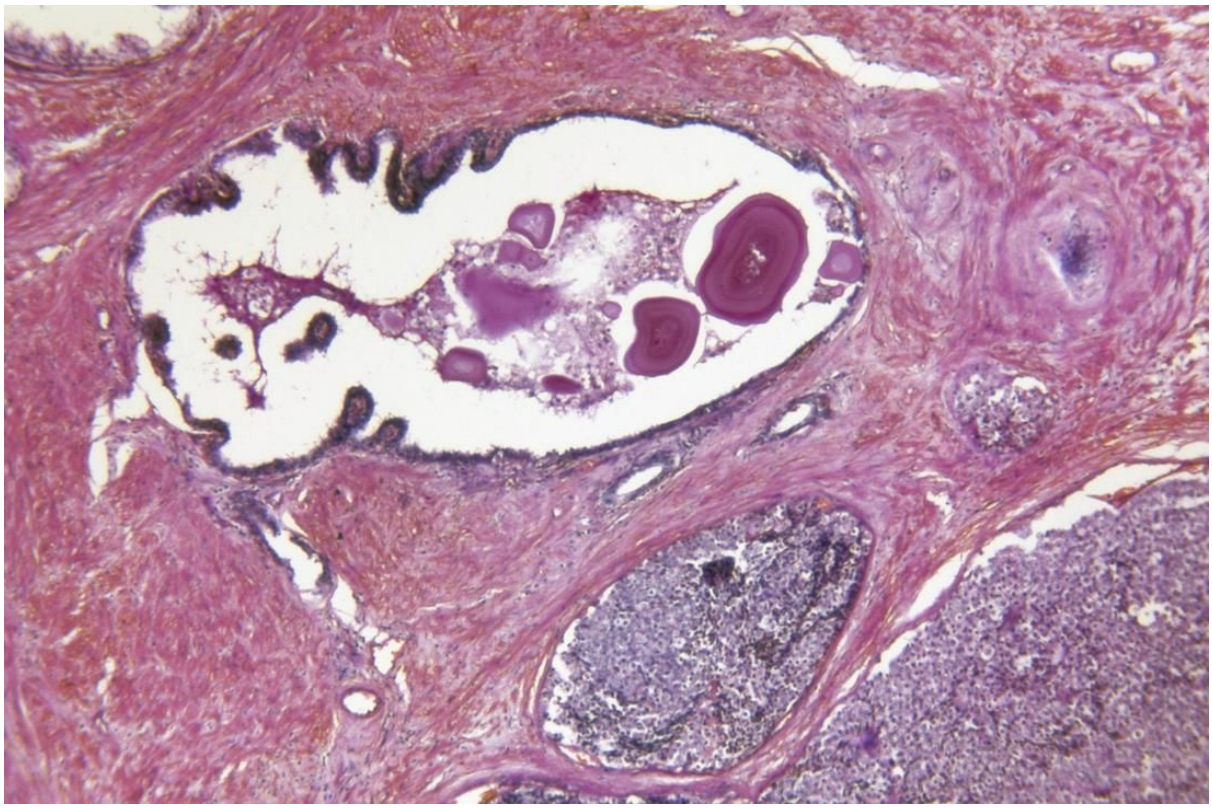
The brain can also develop adenocarcinoma.

DIAGNOSIS

Diagnosis usually begins with an exam that includes a doctor taking a comprehensive medical history of the individual. The doctor will ask questions about symptoms and risk factors, such as smoking.

A number of tests can diagnose adenocarcinoma. Multiple tests may be necessary to confirm the diagnosis.

Biopsy



Adenocarcinoma Tissue Image

This procedure is the removal of a small sample of tissue as shown in the figure to test it for cancerous cells. A biopsy can also provide information about where in the body, cancer

originated from. Some cancers are metastatic cancers or ones that have spread from one area to another.

Imaging scans

A computed tomography (CT) scan is an X-ray that provides three-dimensional images of growth in the body. Doctors sometimes use them to measure change over time and to assess whether treatment is working.

Magnetic resonance imaging (MRI) is another option and uses radio waves to create an image of various parts of the body.

Blood tests

Blood work can measure changes in blood cells that suggest cancer. Chemicals in the blood may also be associated with specific cancers. For example, prostate-specific antigen (PSA) levels change with prostate cancer.

HISTOPATHOLOGICAL DATA COLLECTION

This section elaborates on the data collection process used in this project to obtain whole slide tissue images from an online repository called Genomic Data Commons Data Portal (GDC).

Genomic Data Commons – Overview and Resources

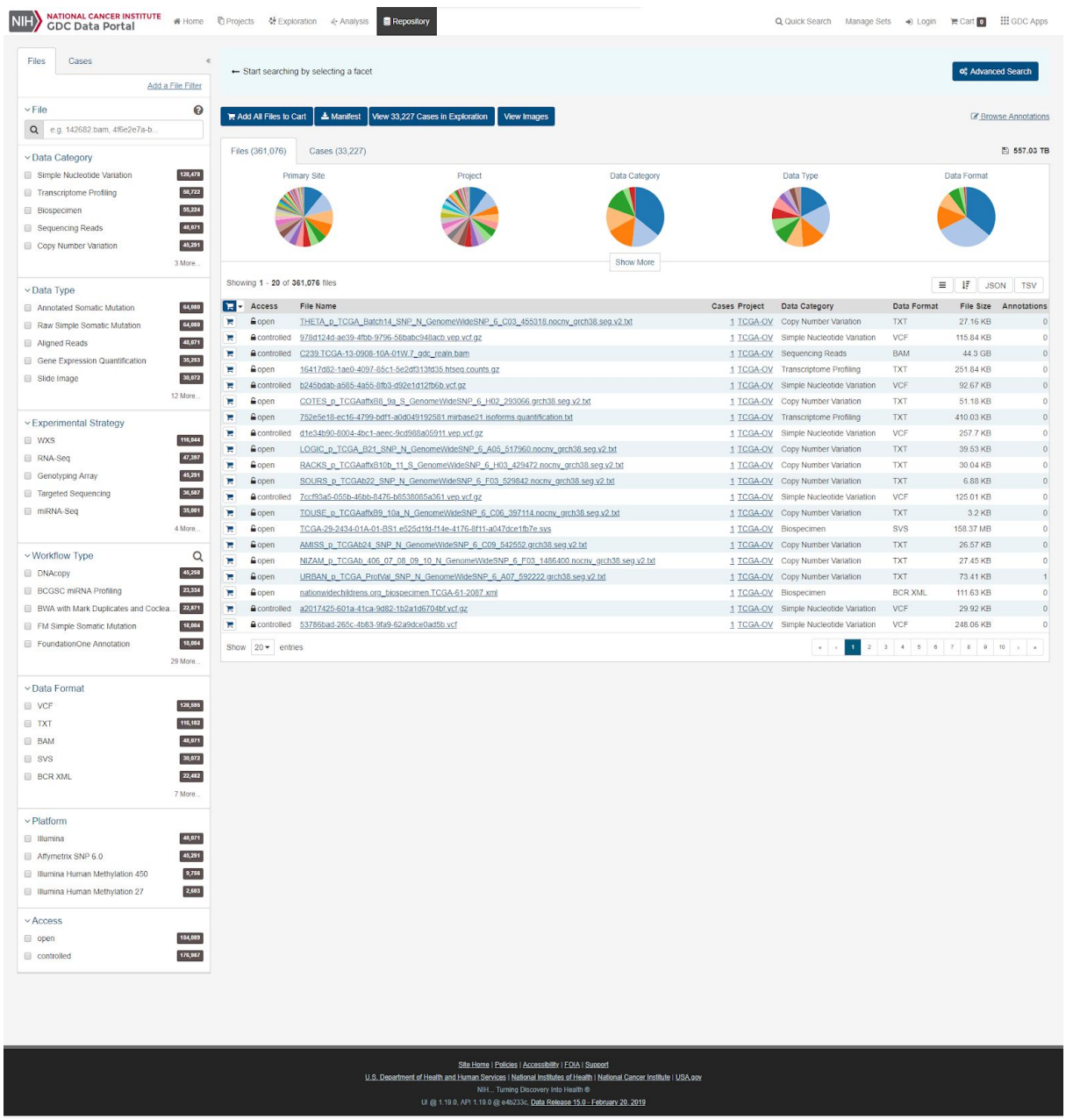
The Genomic Data Commons (GDC) is a research program of the National Cancer Institute (NCI). The mission of the GDC is to provide the cancer research community with a unified data repository that enables data sharing across cancer genomic studies in support of precision medicine.

The National Cancer Institute, part of the National Institutes of Health (NIH), is the federal government's principal agency for cancer research and training. NCI's mission is to lead, conduct, and support cancer research across the nation to advance scientific knowledge and help all people to live longer healthier lives. NCI's scope of work spans a broad spectrum of cancer research across a variety of disciplines and supports research training opportunities at career stages across the academic continuum.

GDC Repository and Client

The GDC repository has a huge collection of genomic data and image database with an easy to access and download portal as shown in the figure below. The GDC provides a standard client-based mechanism in support of high-performance data downloads and submission.

The raw sequence files, typically stored as BAM or FASTQ, make up the bulk of data. The size for a single file can vary greatly depending on the specific analysis; However, some of the whole genome BAM files in The Cancer Genome Atlas (TCGA) reach sizes of 200-300 GB. In such cases, a high-performance data download and submission tool are essential.



GDC Repository

Raw sequence data, stored as BAM files, make up the bulk of data stored at the NCI Genomic Data Commons (GDC). The size of a single file can vary greatly. Most BAM files stored in the GDC are in the 50 MB - 40 GB size range, with some of the whole genome BAM

files reaching sizes of 200-300 GB.

The GDC Data Transfer Tool, a command-line driven application, provides an optimized method of transferring data to and from the GDC and enables resumption of interrupted transfers.

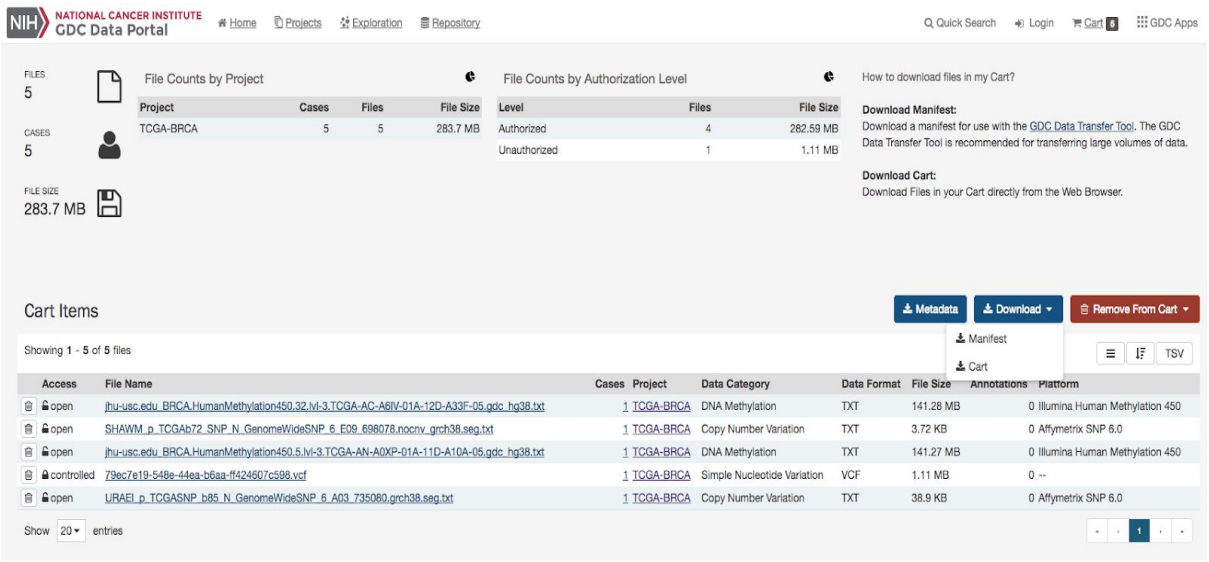
Obtaining a Manifest File for Data Download The GDC Data Transfer Tool supports downloading multiple files listed in a GDC manifest file. Manifest files can be generated and downloaded directly from the GDC Data

Portal:

First, select the data files of interest. Click the Cart button in the row corresponding to the file desired as shown in Figure 4.4. The button will turn green to indicate that the file has been selected.

Adding to Cart

Once all files of interest have been selected, click on the Cart button in the upper right-hand corner. This will bring up the cart page, which provides an overview of all currently selected files. This list of files can be downloaded as a manifest file by clicking on the blue download button and selecting Manifest from the drop down as shown in the figure below.



Downloading Manifest File

After the manifest is downloaded, the command line tool gdc-client can be used to download the data with the simple command: gdc-client download -m manifest.txt. Using gdc-client, the histopathological – whole slide images for 3 types of adenocarcinomas – Stomach, Lung, and prostate are downloaded in the project.

Data Imbalance

Data imbalance is a huge issue in the GDC repository. Since biopsies are taken only after a certain amount of convincing that a lump is a cancer, the number of cancer WSIs are more than the number of normal WSIs. Almost all types of histopathological cancer image datasets have a 5:1 ratio between cancer and normal WSIs in this repository. Overcoming this is one of the big challenges of the training process in the project.

RESEARCH PLAN

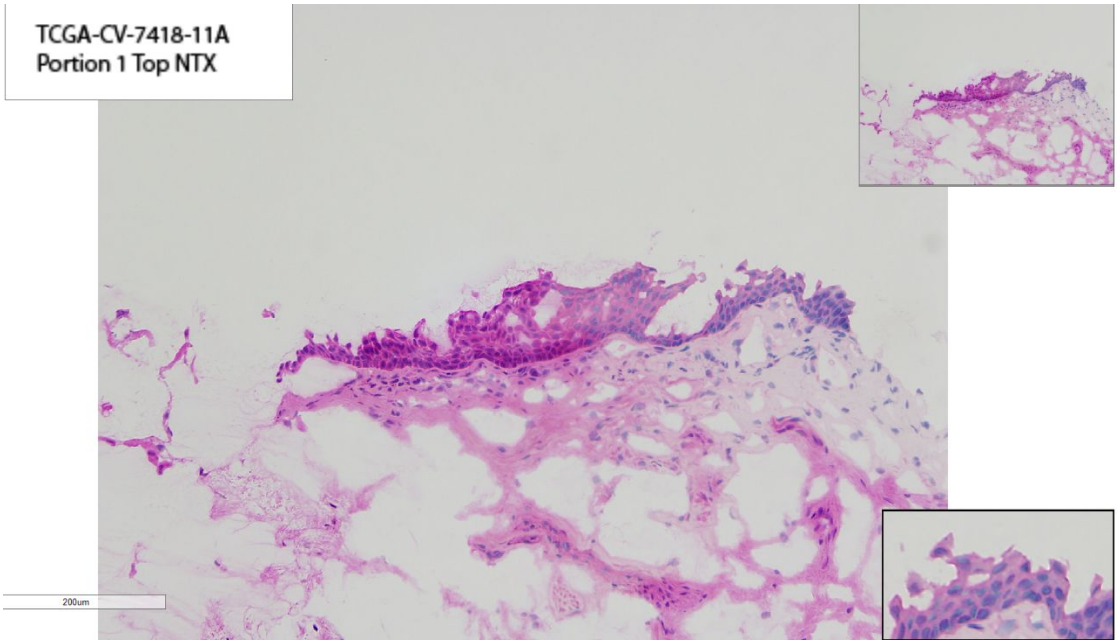
The main objective of the research work is to study the effects of transfer learning in the medical image domain. The specific domain taken for the study is adenocarcinoma detection from WSI of tissues collected during biopsies. The plan has been dissected into four essential parts namely preprocessing, training details, benchmark generation (Experiment I), the study of transfer learning (Experiment II), and Visualization of differences.

PREPROCESSING

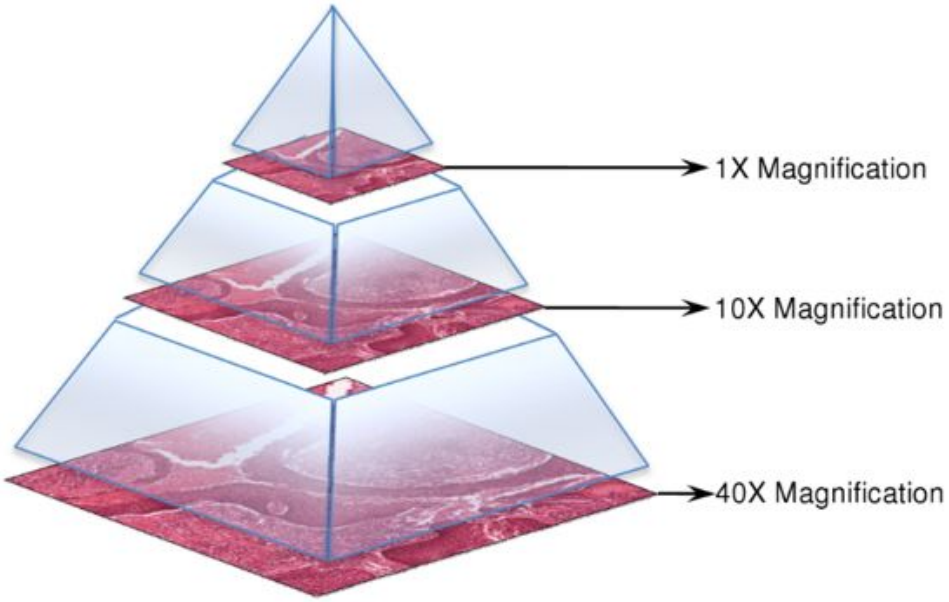
Preprocessing the data obtained from the GDC data portal is done in four stages. First, the data format is analyzed, then, patches are extracted from WSIs. The patches are then scrutinized and filtered for quality purposes and then split into three datasets, each for training, validation and testing, respectively.

Analyzing SVS structure

An SVS file contains virtual slides of different resolutions. The topmost layer also referred to as, layer 0, is always the highest resolution (40X) and the next layer has a thumbnail image of size 1024x768 which is used to represent an example view of the slide. The next few layers contain 20X and 10X resolutions and finally, in the last layer, there is an image of a label containing the relevant information of the slide like the patient number, classification etc. A screenshot of an actual SVS file is provided in the figure below along with a representation of how an SVS file is layered in another figure below that.



View of an SVS File. Top-left – label, bottom-right – 40X magnification of selected portion, top-right – thumbnail, large center image – view of the slide with chosen magnification.



Scanscope Virtual Slides (SVS)

The highest resolution of images can be around 20000*15000 and thus pushing them through a neural network or computing matrix multiplications with them can be a computationally intensive task. Even loading them into the RAM is hard, as the range of file size varies from a few hundred KBs to a few GBs. A simple and proven workaround is to cut the high-resolution whole slide image into patches and train on those patches.

This workaround is applicable on one condition. The class of the classified image should be applicable to the patches as well, for example, in this project, the cancerous cells of adenocarcinoma of the chosen organs are all spread throughout the tissue and thus all patches of cancerous tissue are also cancerous.

OpenSlide Patch Extraction

The Python interface for OpenSlide library provides functionalities to extract regions from a layer in the SVS file. One such functionality called the read_region is used in this project to extract patches of 512x512 size. Each patch is an 8-bit RGB image.

```
osr = OpenSlide.OpenSlide(<svs file path>)

patch = osr.read_region(location=(<x>, <y>), level=<layer number>, size=(<width>, <height>))
```

Patch Analysis and Filtering

After a patch has been extracted, and converting it into RGB from RGBA and then the validity of the patch has to be ascertained before appending it to the dataset. The validity is determined by 2 conditions,

- The patch should not be blank, as it would serve no purpose for detection
- At least 50% of the patch should contain the tissue sample, as there would not be enough tissue to detect (e.g. corners of the tissues beyond which the slide is empty)

```
Condition 1: patch.getbbox() == None => Blank slide

Condition 2: (from PIL library) ImageStat.Stat(Image.eval(patch, lambda x: 1 if x >= 210)) => Gives number of pixels the just white.
```

Data Splits

The validated patches are placed into train, validation and test folders based on the number of slides processed until then. 70% of the slides are placed in the training folder and 15% of the slides are placed under the validation and testing folder each.

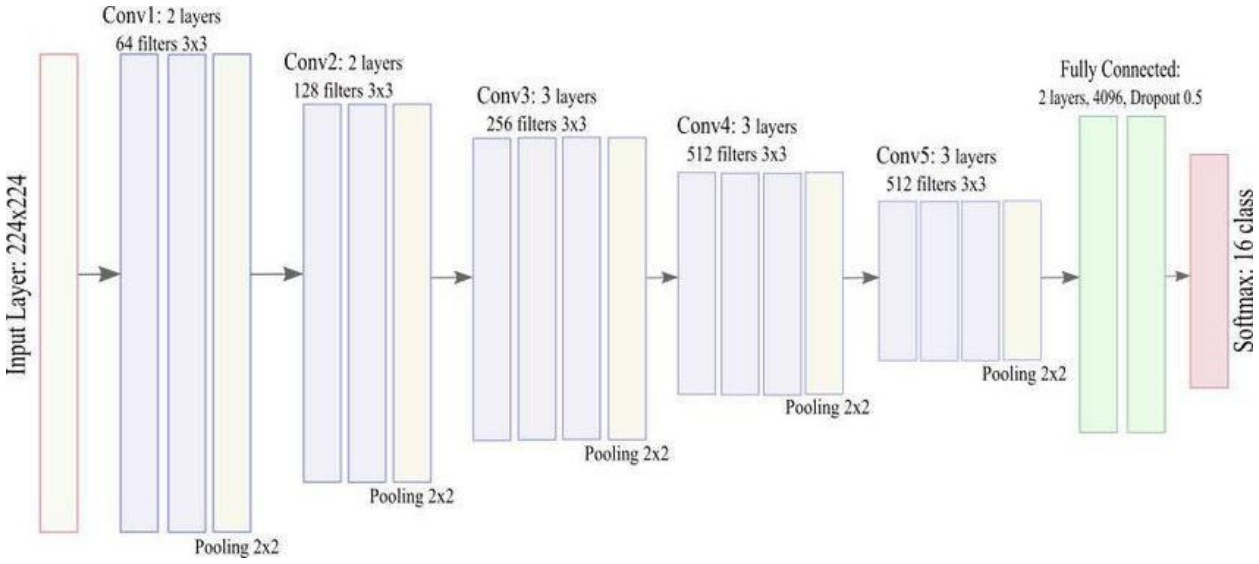
TRAINING DETAILS

The steps involved in training the deep learning models for the project are described in this section.

Model Architecture

Throughout this project, the CNN based VGG-16 model architecture is used as shown in the figure below.

The model contains 13 convolutional layers, 2 fully connected layers and one softmax classifier. For this project, the classification layer is replaced with a softmax with 2 classes, cancer and normal, instead of 16. ImageNet pre-trained weights are loaded as initial weights.



VGG-16 Architecture - Each Convs+Pooling Layer is named as a Block

Input > B1 > B2 > B3 > B4 > B5 > Fully connected layers > Softmax

Training Configurations

The model is created with the following configurations after a manual trial and error grid search.

- **Adam Optimizer**
    - The initial **learning rate** is low compared to popularly preferred rates. Due to the data imbalance, higher learning rate can make the training unstable and fluctuate. The range of initial learning rate values used in this project are from 10^{-5} to 10^{-4}
    - **Weight decay** is used to provide L2 regularization effect for the training. The range of values used for weight decay in this project are from 10^{-4} to 10^{-2}
    - **Betas,** i.e. beta values for the Adam optimizer are not changed from their default values 0.9 and 0.999.
- **Learning Rate Scheduler**
    - **ReduceLROnPlateau** is used for scheduled decay of learning rate when the training stagnates because of the high learning rate.
    - The scheduler has the patience of 2 epochs, i.e. the learning rate is changed after 2 continuous epochs of no improvement in loss value of the training
    - The decay rate is set to 0.2, i.e. when the decay takes place, the new learning rate is 0.2 times the old learning rate
- **Dropout**
    - Dropout regularization is applied at the fully connected layers
    - Dropout rates are in the range [0.3, 0.5] for the training
- 15 **Epochs** for training, because each epoch can take several hours due to the huge number of images, ~1 Million
- **Data Augmentation** is applied to make the system robust. Pytorch applies image transforms at the input layer.

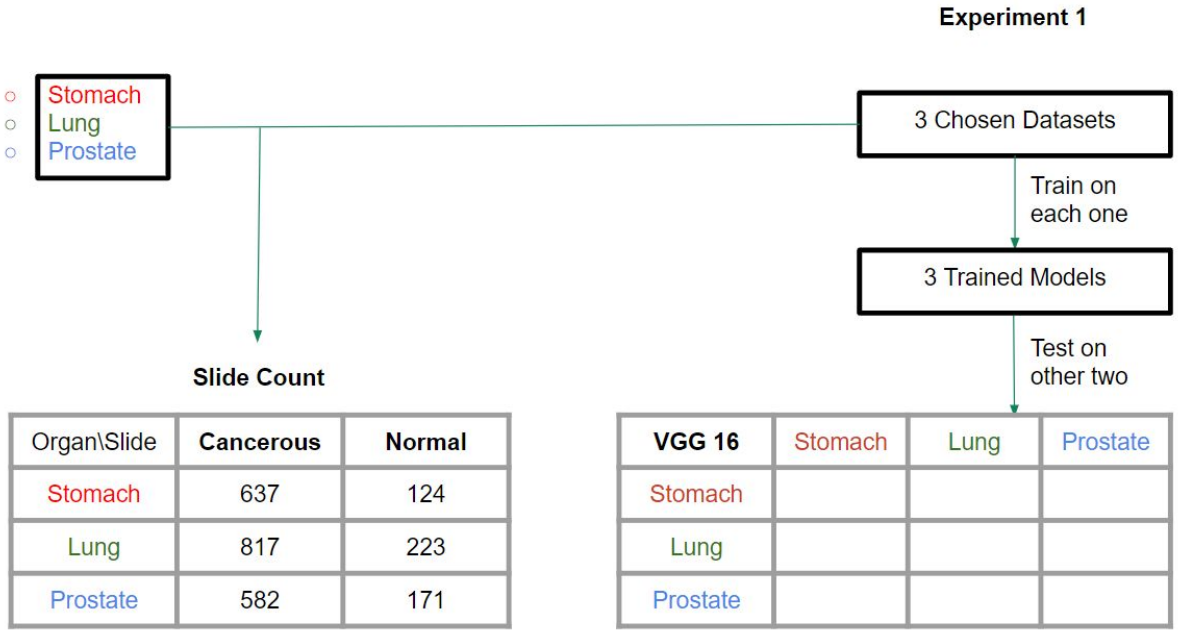
Metrics

The model calculates the following metrics after each epoch.

- Accuracy
- Loss
- Precision
- Recall
- F2 score (F measure with beta=2)

EXPERIMENT I

The first experiment of the project aims at establishing a benchmark of performance measures between three models trained on adenocarcinoma of three different organs, namely the Lung, Prostate and Stomach. A flow diagram of the experiment is provided below in the figure below. The steps of this experiment are presented in this section.



Organ\Slide	Cancerous	Normal
Stomach	637	124
Lung	817	223
Prostate	582	171

VGG 16	Stomach	Lung	Prostate
Stomach			
Lung			
Prostate			

The Flow of Experiment 1

Training

The experiment starts by training three VGG-16 models for each organ. Pretrained ImageNet weights are used as initial weights but no layer is frozen, thus all features are trained based on the new data. The least error model with highest F2 scores is saved for each organ by saving the model states along with logs of the training process regarding the metrics.

Cross Testing

When training is complete, each organ model is cross tested on the other two organs' test splits, i.e. the Stomach model evaluates on both the Lung and Prostate test data. At the end of this, the benchmark is filled with the test results.

EXPERIMENT II

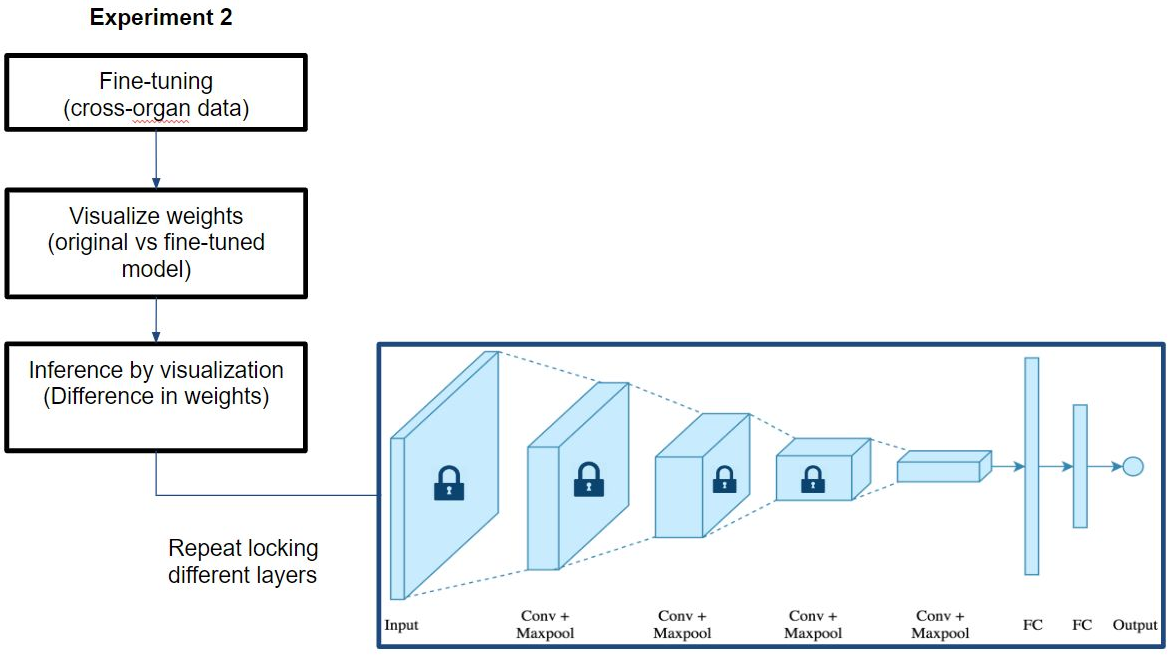
The second experiment of the project has two objectives.

It aims to fine-tune the trained organ models with other organ data. It also analyzes and compares the performance of above-mentioned models initialized with pre-trained weights from the same domain with models obtained by pretraining from ImageNet weights.

The second objective of this experiment is to identify the number of generalizable layers present in the network by freezing layers incrementally and learning. This section explains the implementation of this project to transfer histopathological features from one model to another. The flow of the experiment is shown in the figure below.

Fine-Tuning Cross-Organ Data

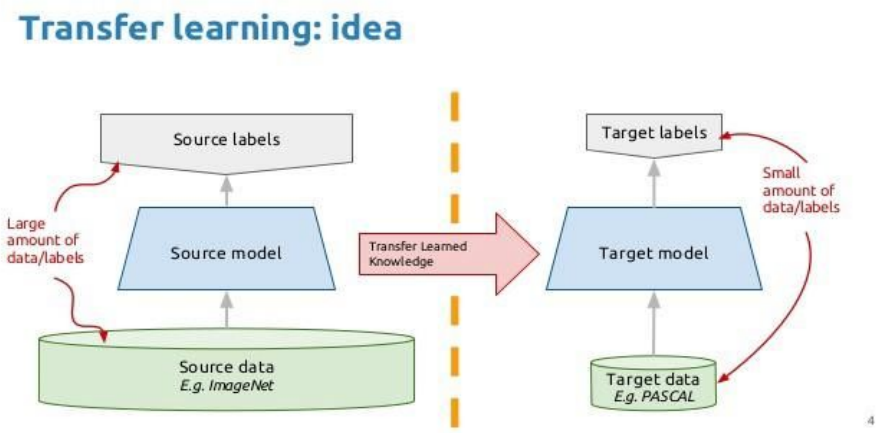
Fine-tuning is the process of loading weights/features of a pre-trained model into another model as initial weights and training the new model. Usually only the last few layers, the fully connected layers of the model are trained and other layers are frozen as shown in the figure below. But in this experiment, the layers are incrementally frozen to identify generalizable layers.



Flow of Experiment 2

Original Vs Fine-Tuned Weights (Visualization and Inference)

The second step in the experiment is to visualize the weights of the model before and after fine-tuning to identify key differences that Adenocarcinoma of different organs present. The visual differences obtained in this step provide insight on what aspects of the weights change and how generalizable are these weights based on the extent of the changes.



The Flow of Transfer Learning

VISUALIZATION OF DIFFERENCES

Visualization of CNN models can be done in several different ways to get different insights. The visualizations that have been included as a part of this project include,

- Gradient visualization with vanilla backpropagation
- Gradient visualization with guided backpropagation
- Gradient visualization with the saliency map
- Gradient-weighted class activation mapping - Grad-CAM
- Guided, gradient-weighted class activation mapping
- Smooth grad
- CNN filter visualization
- Inverted image representations
- Deep dream
- The class-specific image generation
- TSNE
- Filter Visualization
- Probability Heatmap

To understand what each visualization tries to convey, please see the following Github repository for CNN visualizations - Pytorch-CNN-Visualizations. Comparing such visualizations with fine-tuned models can give insight into the difference between learning from ImageNet and Learning from Adenocarcinoma itself.

The probability heatmap and TSNE will be explained in the results.

CHALLENGES, RESULTS AND CONCLUSIONS

EXPERIMENT I

model\test	Stomach	Lung	Prostate
Lung	87.74	93.93	81.63
Prostate	80.29	77.03	94.17

The accuracy benchmark for comparison with transfer learning
Only results on test split are shown

Challenges

The training for the Stomach based model is incomplete due to overwhelming data imbalance of a little more than 10:1 ratio between cancer and normal patches. Perceivable reasons for unexpected data imbalance explosion include,

- The pixel resolution of WSIs, for the same 40x magnification, are different, ranging from 15000-50000 pixels for both width and breadth of the image. Thus, each slide gives a varying number of patches during patch extraction. Enforcing a limit to the number of patches seems like a good solution but each slide varies in the number of valid patches it can give, i.e. without white patches and containing at least 50% tissue content
- Regularization did not have any effect. Batch normalization, weight decay and dropout regularization did not stop over-fitting
- The weighted loss did not have any effect on the over-fitting

- Cross-validation was not an option due to the large training time.

Results

Results provide us with a decent benchmark for comparing with fine-tuned models.

Conclusions

Decent benchmark models have been obtained for both prostate and lung based adenocarcinoma detection. Stomach based benchmark model could not be obtained due to huge data imbalance.

EXPERIMENT II

Accuracy Values for Freezing Until Block B(x)

Model	Phase	B1	B2	B3	B4	B5
Lung Model on Prostate Data	Training	93.87	96.19	91.54	80.75	66.18
	Validation	85.12	88.28	87.82	78.59	65.98
	Test	86.65	91.54	89.44	73.34	60.71
Prostate Model on Lung Data	Training	94.54	95.34	95.33	92.81	83.58
	Validation	94.16	93.67	94.73	93.17	84.81
	Test	91.10	91.81	92.27	89.56	82.28

Results

These results signify that features until block 3 of the lung model and block 2 for the prostate model are transferable. Chosen based on accuracy, bias and variance.

Conclusions

Fine-tuned models for both prostate and lung model have been obtained and they perform on par with benchmark models. The extent to which features are generalizable was also tested.

VISUALIZATIONS

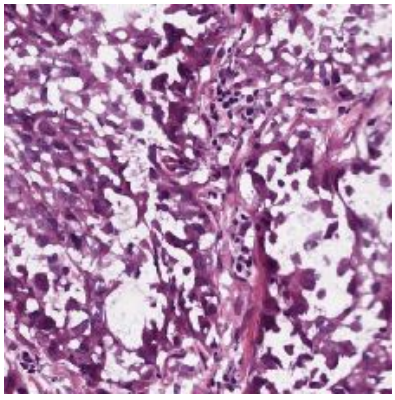
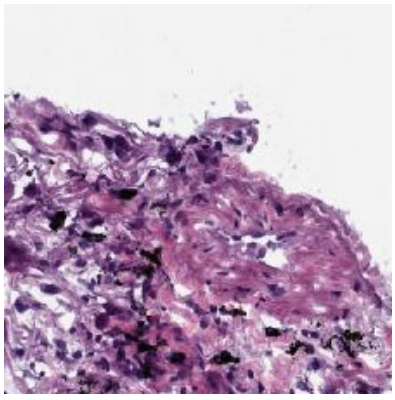
Results

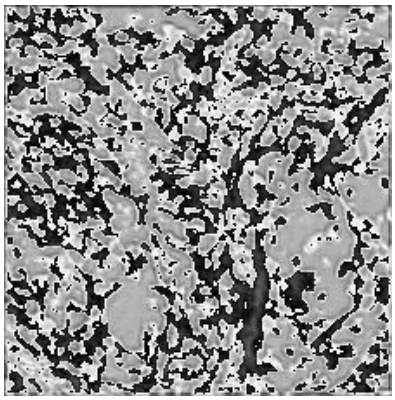
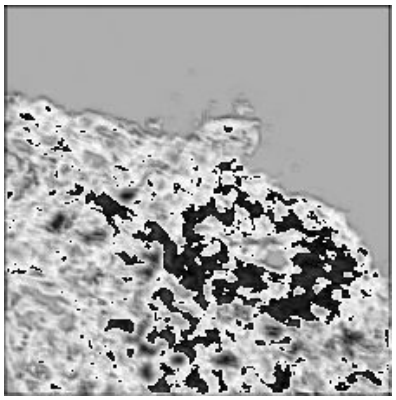
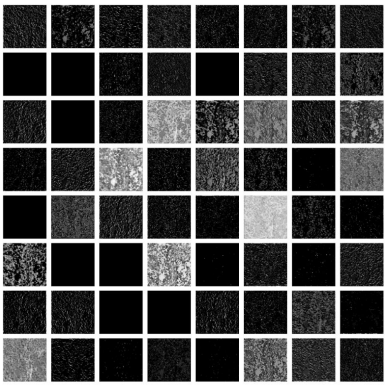
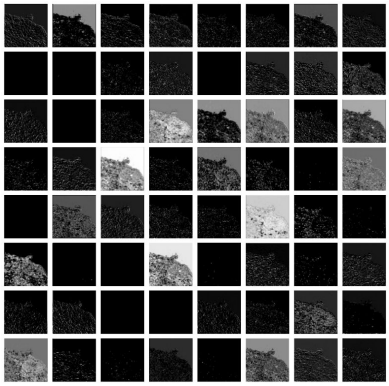
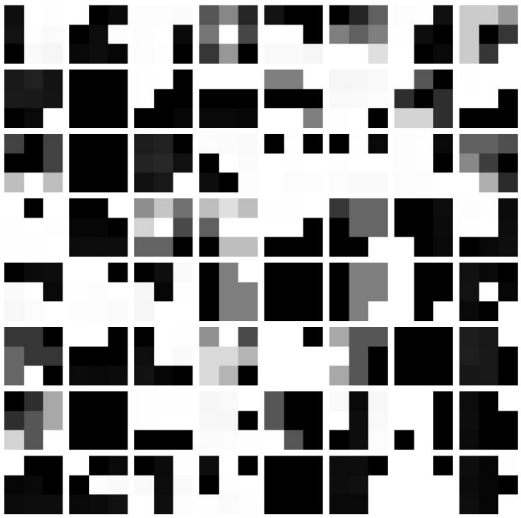
The visualization of any model has certain procedures associated with it. Using one patch each for cancer and normal from both lung and prostate based adenocarcinoma, the procedure is explained below.

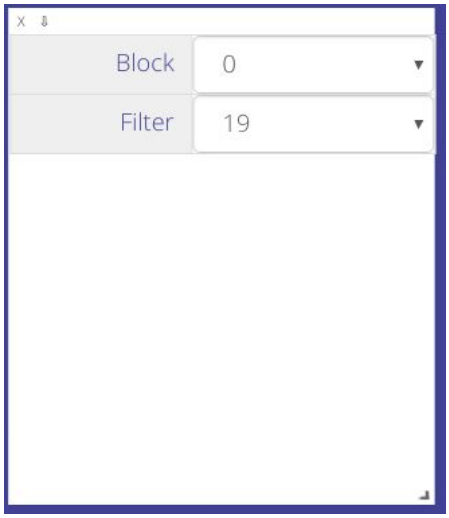
Visualizing Layers

The most common first visualization is the visualization of the filters of layers, to identify what effect they have on the input. The file visdom-vgg.py provides this functionality. A few examples are shown below. Using this first step, we can find which filters are interesting.

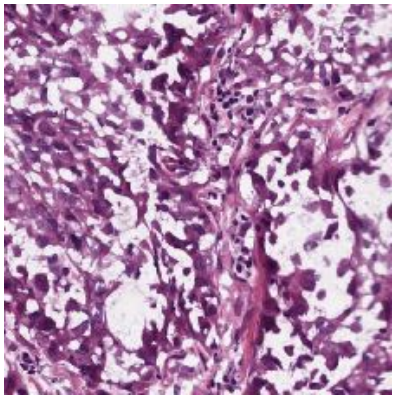
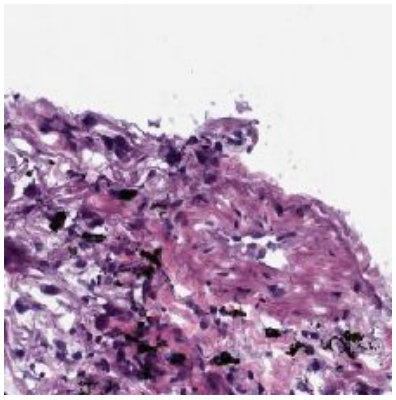
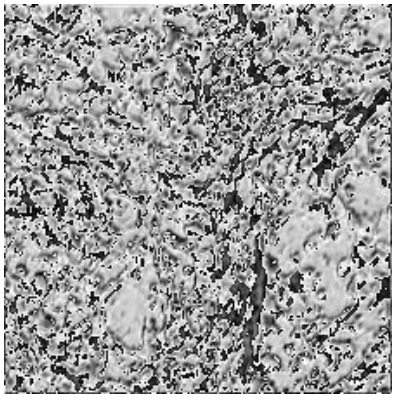
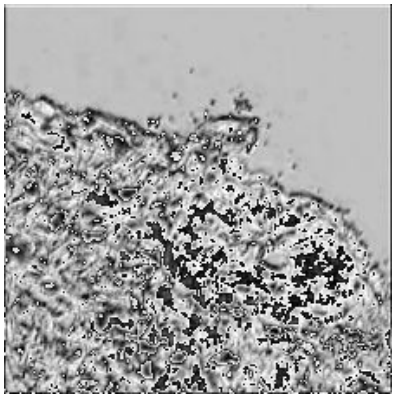
Lung Model on Lung Data

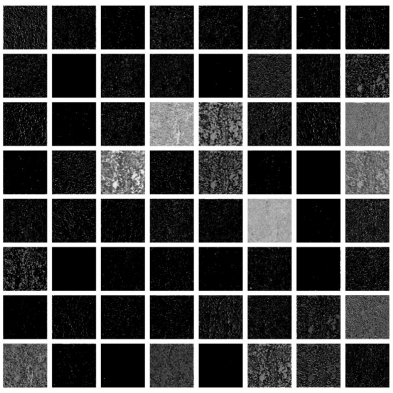
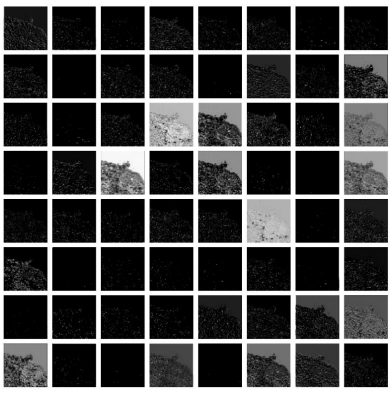
	Cancer	Normal
Original		

Results of Filter 19 From Block 0 Layer 2		
Results of all the filters in the Block 0 Layer 2		
Components of Filter 19		

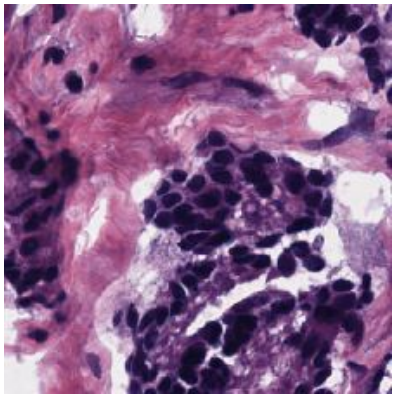
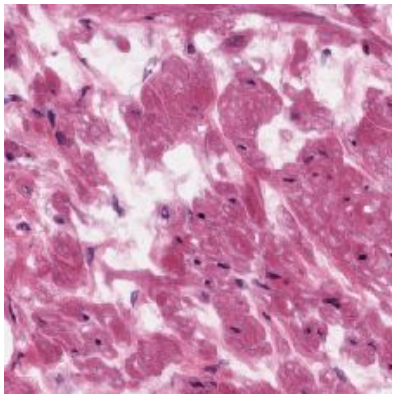
Controller to change block and filter	

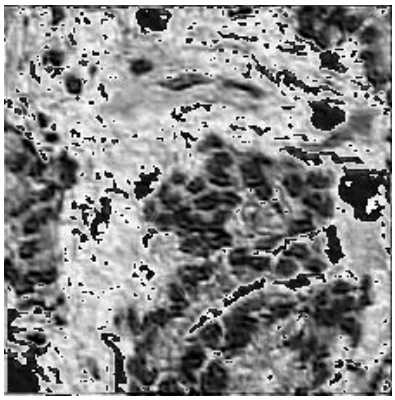
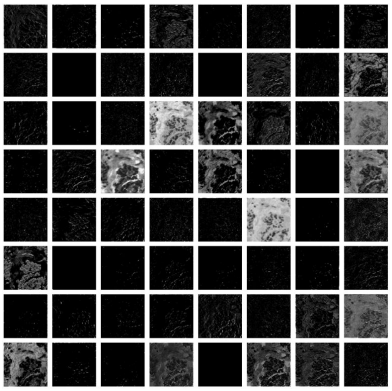
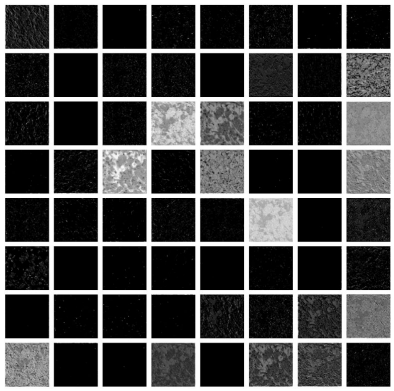
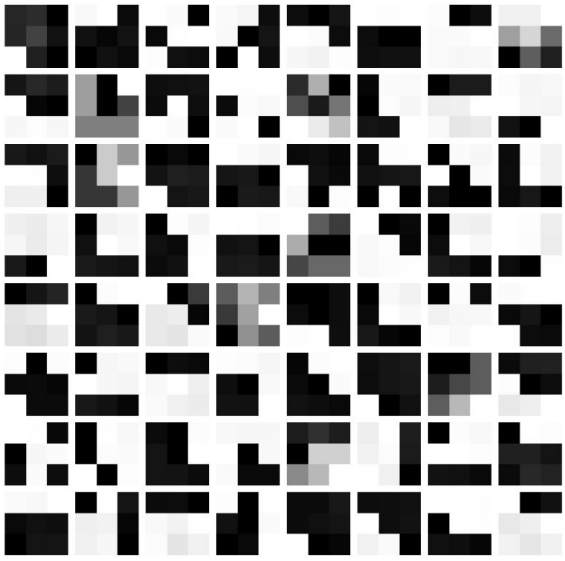
Prostate Model on Lung Data

	Cancer	**Normal**
Original		
Results of Filter 19 From Block 0 Layer 2		

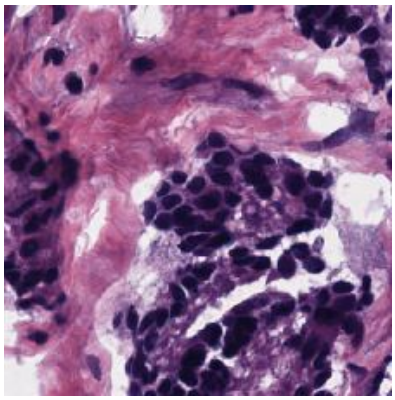
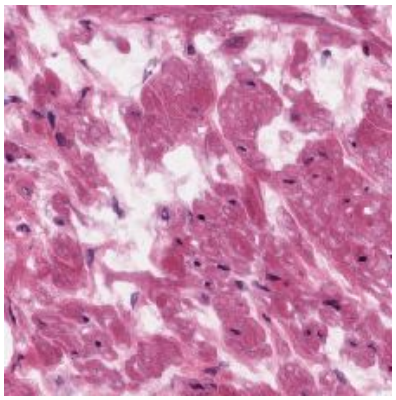
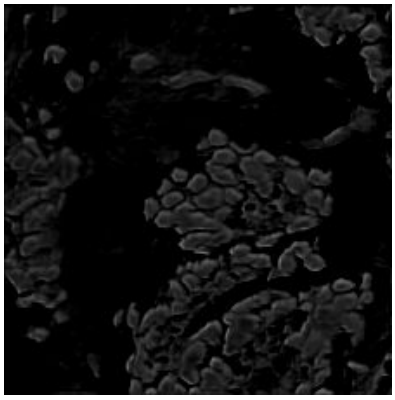
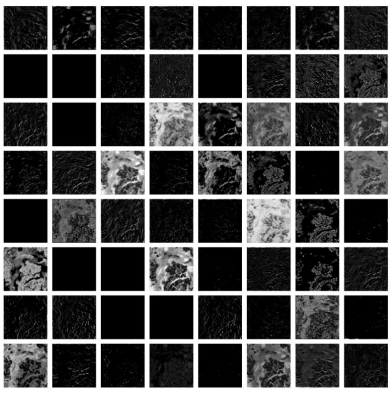
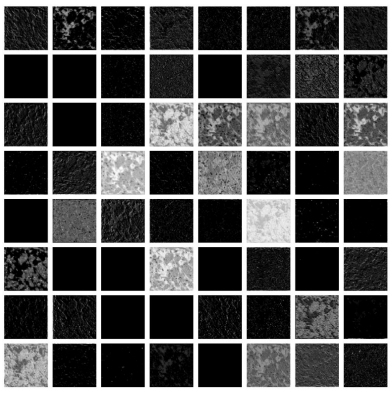
Results of all the filters in the Block 0 Layer 2		
Components of Filter 19		

Prostate Model on Prostate Data

	Cancer	**Normal**
Original		

Results of Filter 31 From Block 0 Layer 2		
Results of all the filters in the Block 0 Layer 2		
Components of Filter 31		

Lung Model on Prostate Data

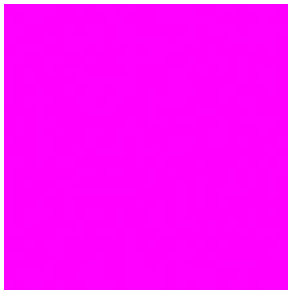
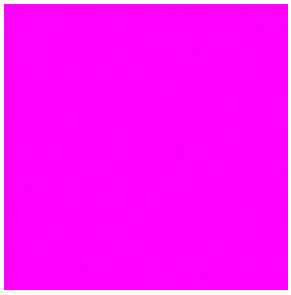
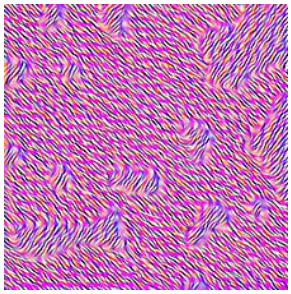
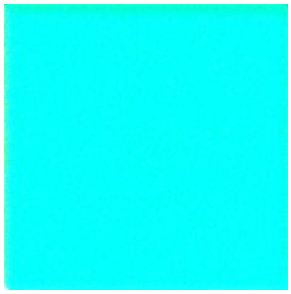
	Cancer	Normal
Original		
Results of Filter 38 From Block 0 Layer 2		
Results of all the filters in the Block 0 Layer 2		

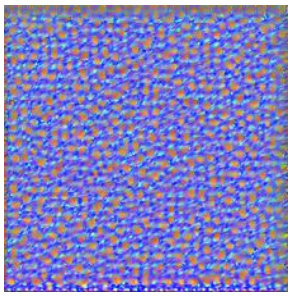
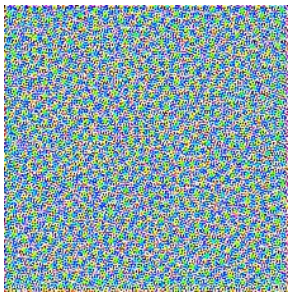
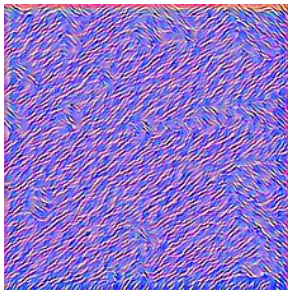
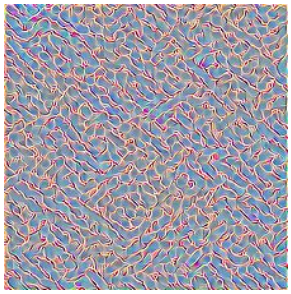
Components of Filter 19	

Visualizing Filter Expectations

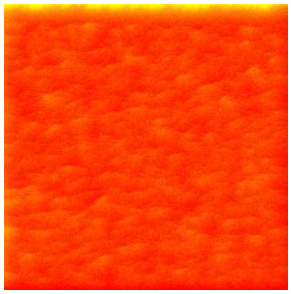
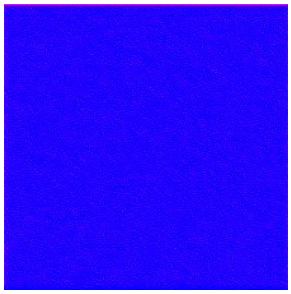
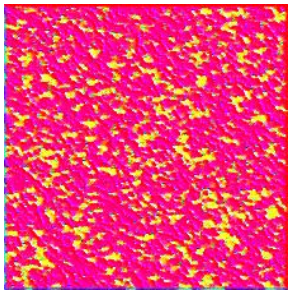
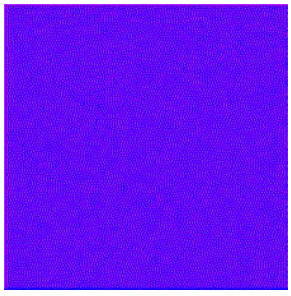
To find out what a filter expects is a form a visualization that can help understand a model. The cnn_layer_visualization.py finds out what a filter expects by starting with a noise and iterating to reduce the loss. The following examples show what some interesting filters expect.

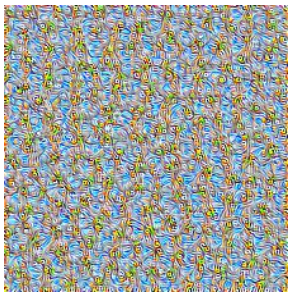
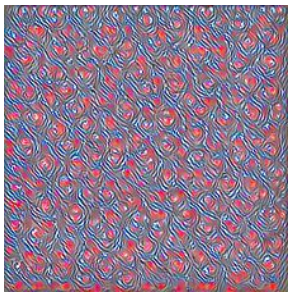
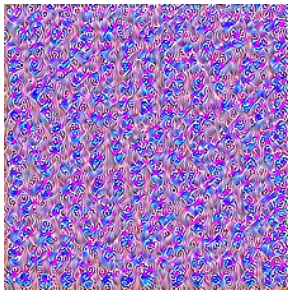
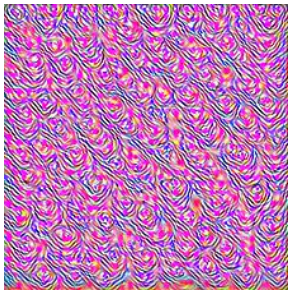
Lung Data

	Lung Model		**Prostate Model with Lung Data**
Block 0 Layer 2 Filter 19		**Block 0 Layer 2 Filter 19**	
Block 0 Layer 2 Filter 56		**Block 0 Layer 2 Filter 61**	

Block 2 Layer 14 Filter 13		**Block 2 Layer 14 Filter 112**	
Block 2 Layer 14 Filter 214		**Block 2 Layer 14 Filter 214**	

Prostate Data

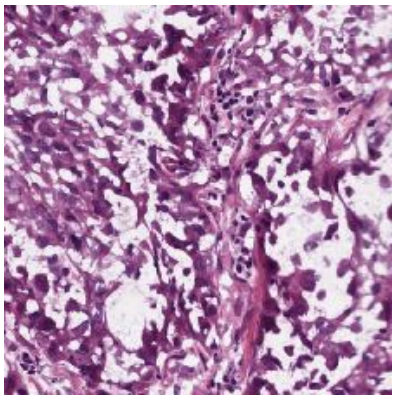
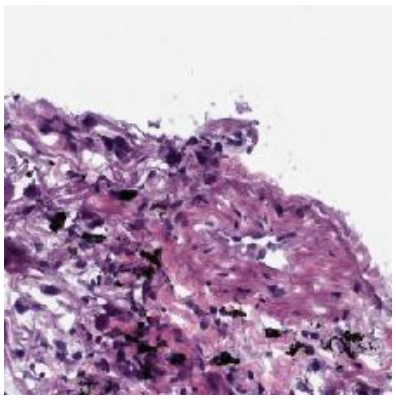
	Prostate Model		**Lung Model with Prostate Data**
Block 0 Layer 2 Filter 15		**Block 0 Layer 2 Filter 38**	
Block 0 Layer 2 Filter 31		**Block 0 Layer 2 Filter 40**	

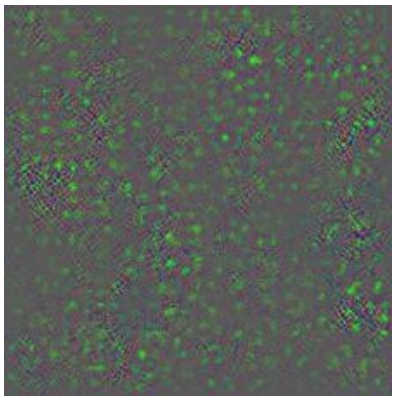
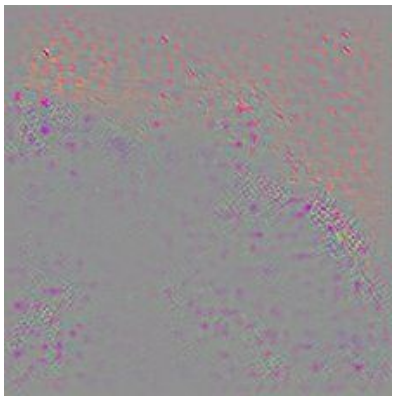
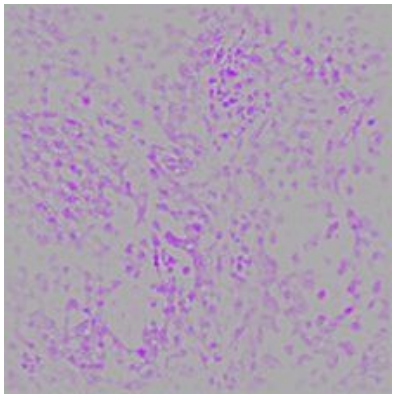
Block 3 Layer 21 Filter 178		**Block 3 Layer 21 Filter 117**	
Block 3 Layer 21 Filter 253		**Block 3 Layer 21 Filter 511**	

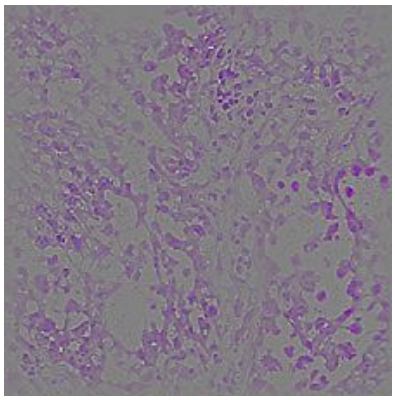
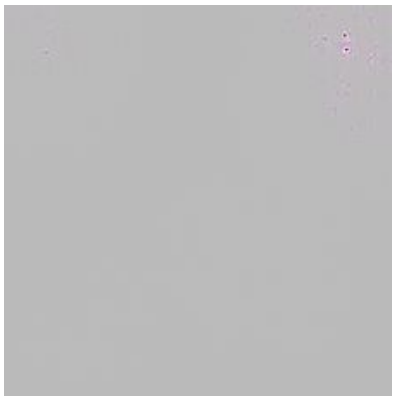
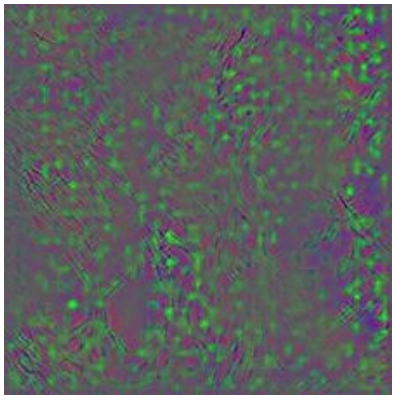
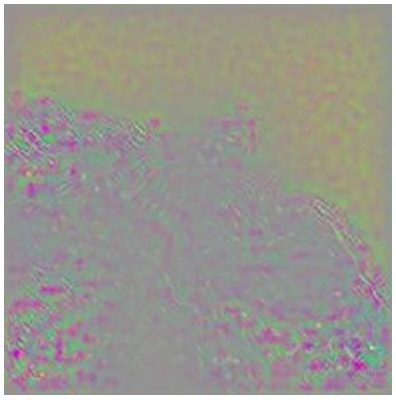
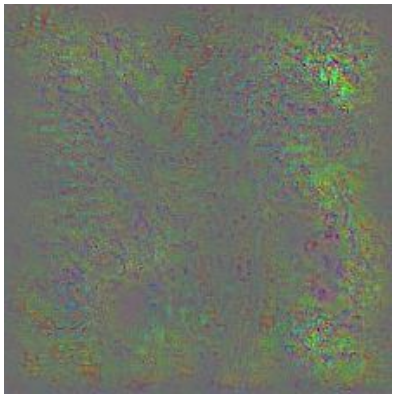
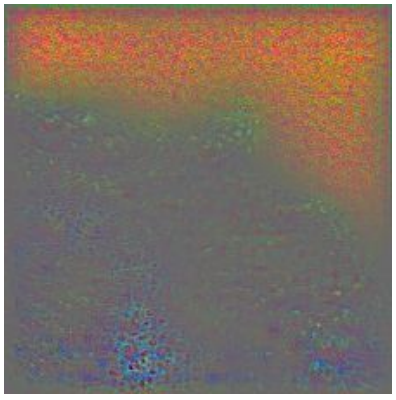
Backprop

The visualization created by vanilla_backprop.py, guided_backprop.py and smooth_grad.py are used to identify regions of interest for the model in the input image using backpropagation of gradients.

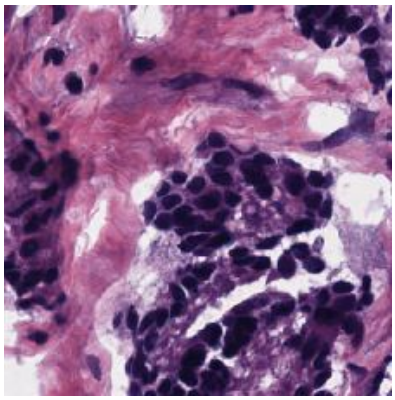
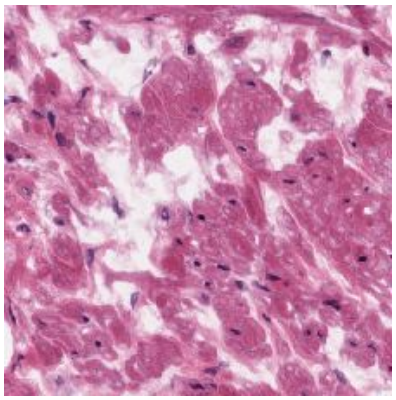
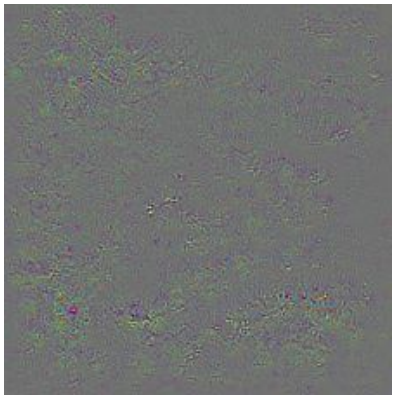
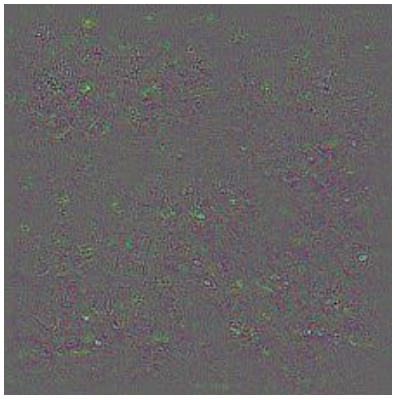
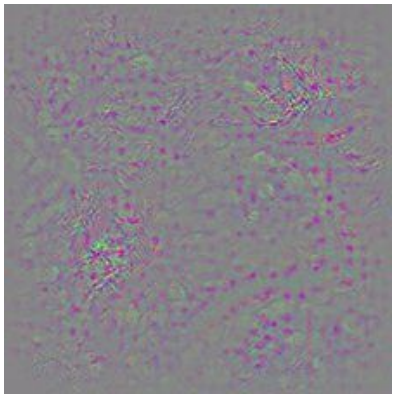
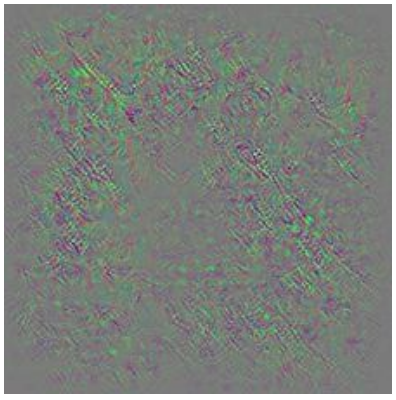
Lung Data

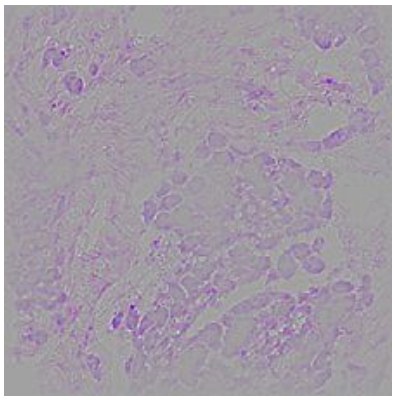
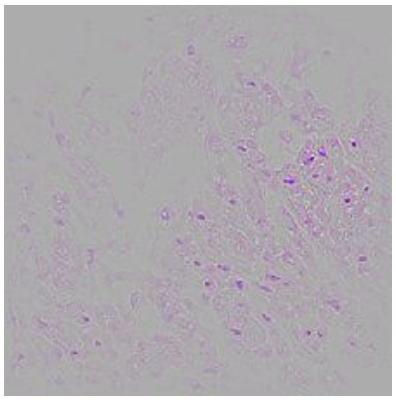
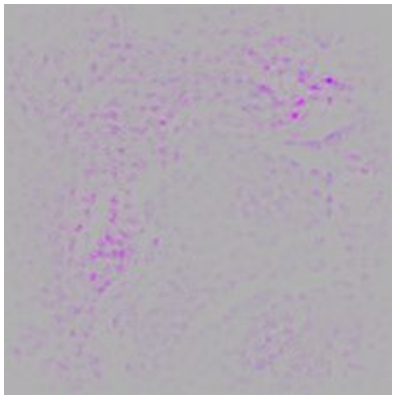
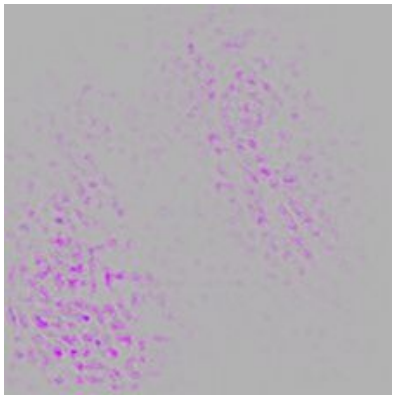
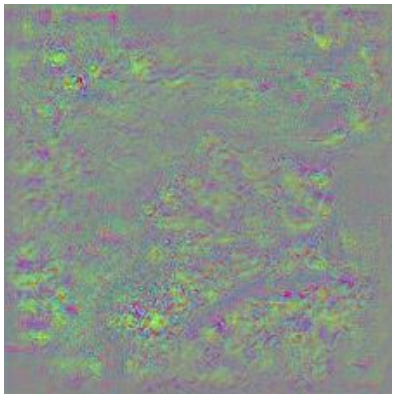
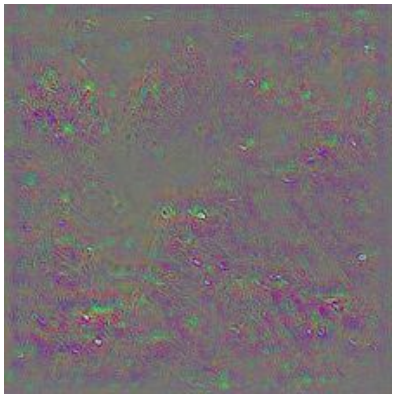
	Cancer	**Normal**
Original		

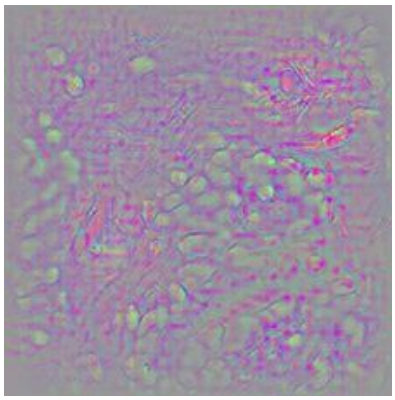
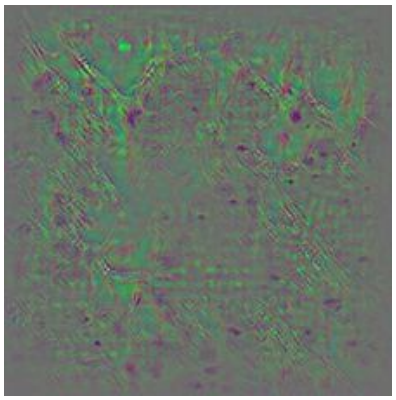
Lung Model Vanilla Backprop		
Prostate Model on Lung Data Vanilla Backprop		
Lung Model Guided Backprop		

Prostate Model on Lung Data Guided Backprop		
Lung Model Smooth Grad		
Prostate Model on Lung Data Smooth Grad		

Prostate Data

	Cancer	Normal
Original		
Prostate Model Vanilla Backprop		
Lung Model on Prostate Data Vanilla Backprop		

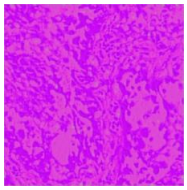
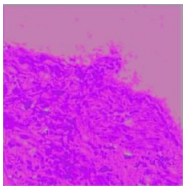
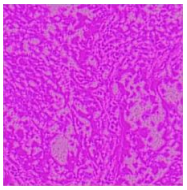
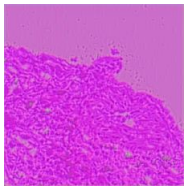
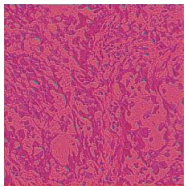
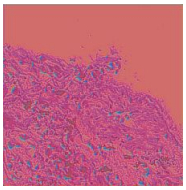
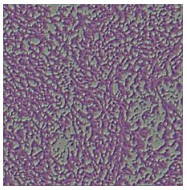
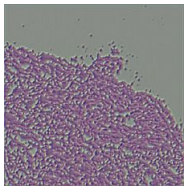
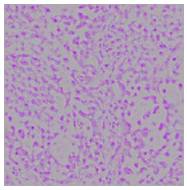
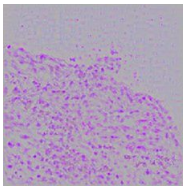
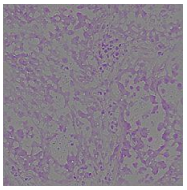
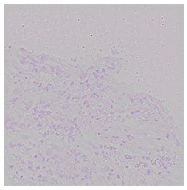
Prostate Model Guided Backprop		
Lung Model on Prostate Data Guided Backprop		
Prostate Model Smooth Grad		

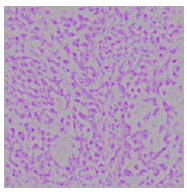
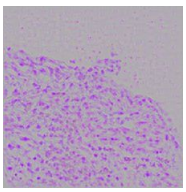
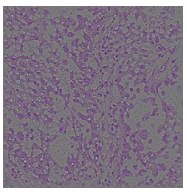
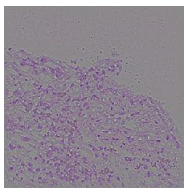
Lung Model on Prostate Data Smooth Grad		

Filter - Guided Backprop

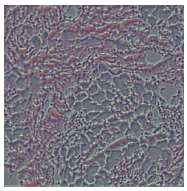
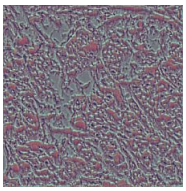
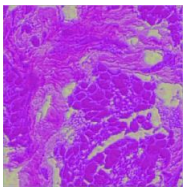
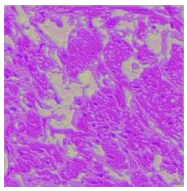
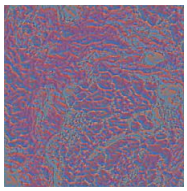
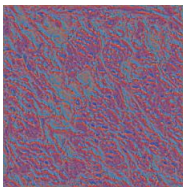
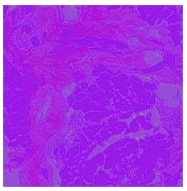
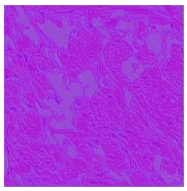
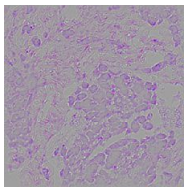
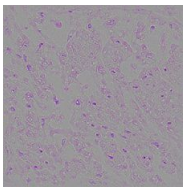
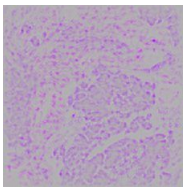
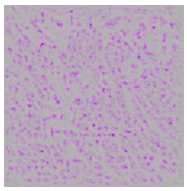
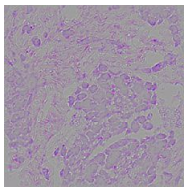
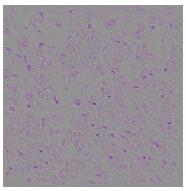
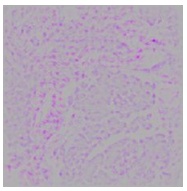
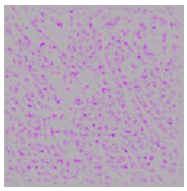
The code layer_activation_with_guided_backprop.py, provides a CAM for the filter on the image based on guided backpropagation.

Lung Data

	Lung Model Cancer	**Lung Model Normal**		**Prostate Model with Lung Data Cancer**	**Prostate Model with Lung Data Normal**
Block 0 Layer 2 Filter 19			**Block 0 Layer 2 Filter 19**		
Block 0 Layer 2 Filter 56			**Block 0 Layer 2 Filter 61**		
Block 2 Layer 14 Filter 13			**Block 2 Layer 14 Filter 112**		

Block 2 Layer 14 Filter 214			**Block 2 Layer 14 Filter 214**		

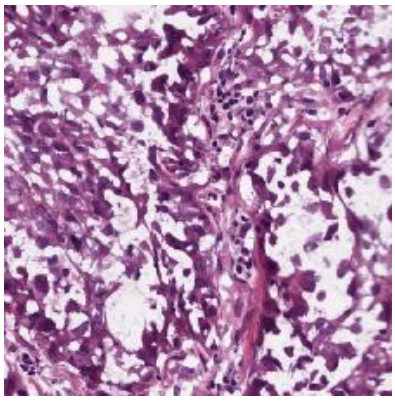
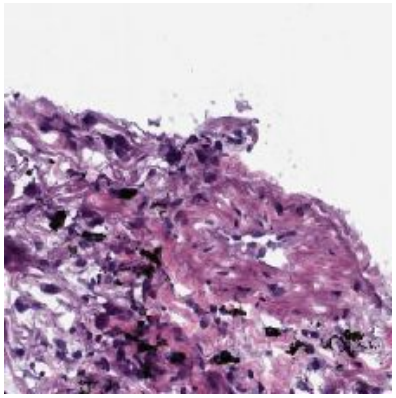
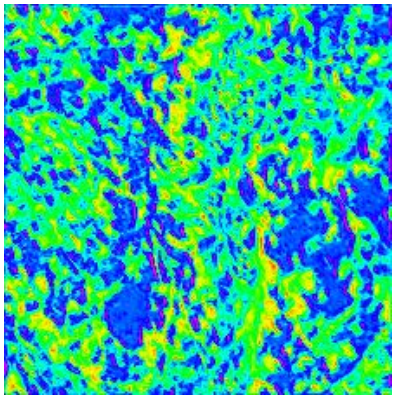
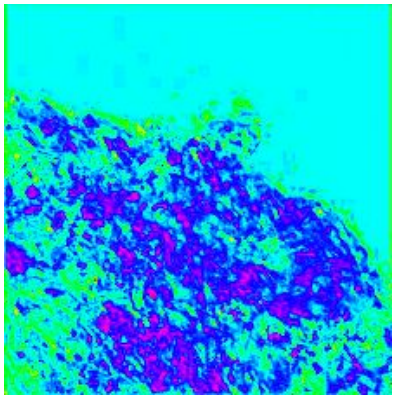
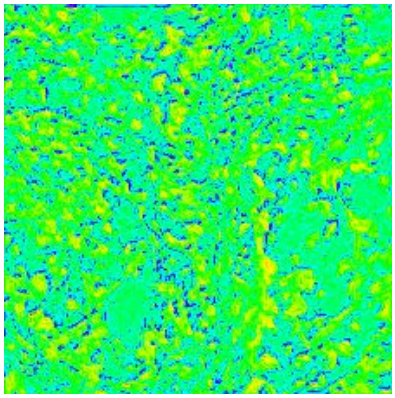
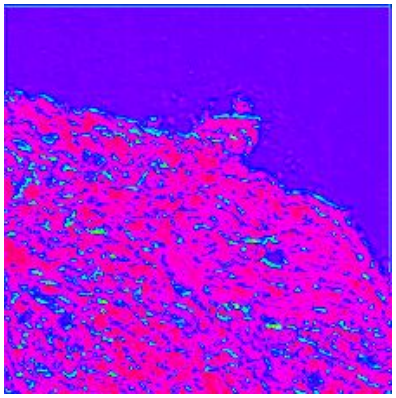
Prostate Data

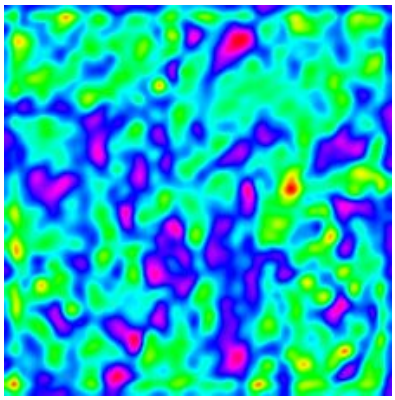
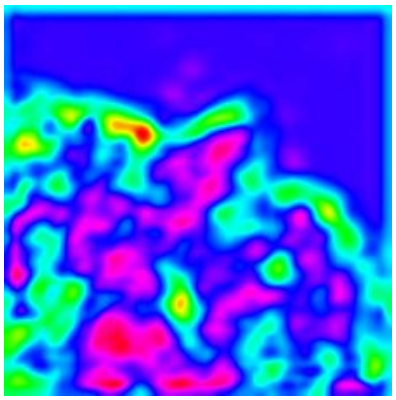
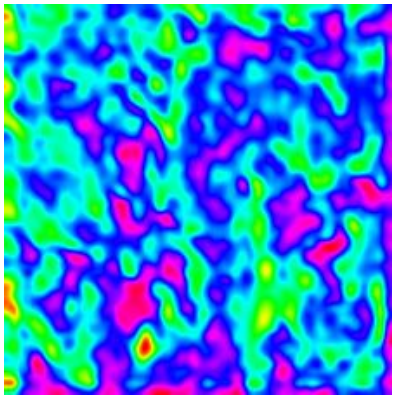
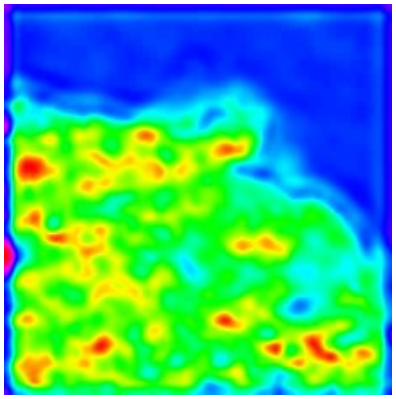
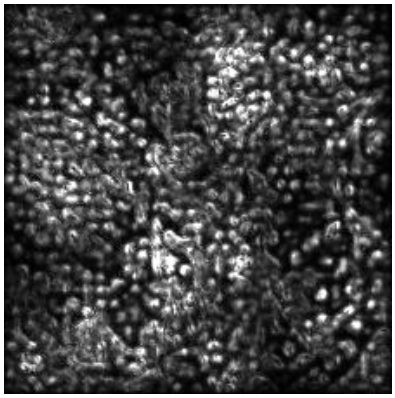
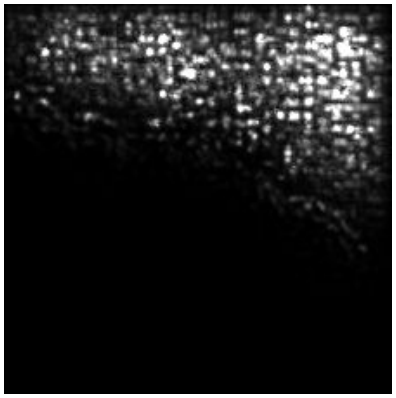
	Prostate Model Cancer	**Prostate Model Normal**		**Lung Model with Prostate Data Cancer**	**Lung Model with Prostate Data Normal**
Block 0 Layer 2 Filter 15			**Block 0 Layer 2 Filter 38**		
Block 0 Layer 2 Filter 31			**Block 0 Layer 2 Filter 40**		
Block 3 Layer 21 Filter 178			**Block 3 Layer 21 Filter 117**		
Block 3 Layer 21 Filter 253			**Block 3 Layer 21 Filter 511**		

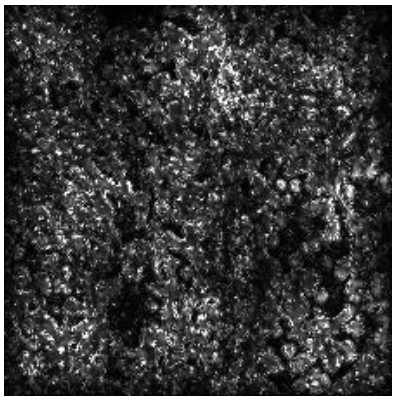
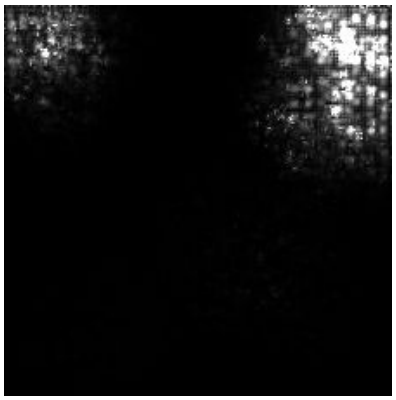
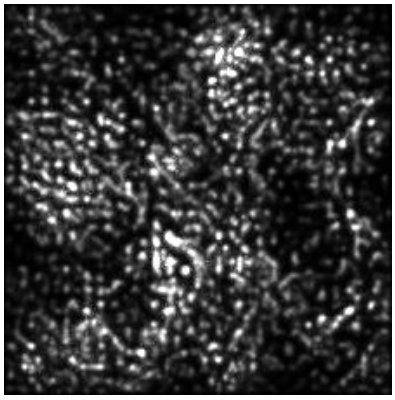
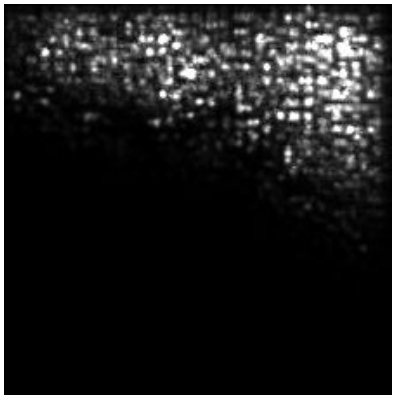
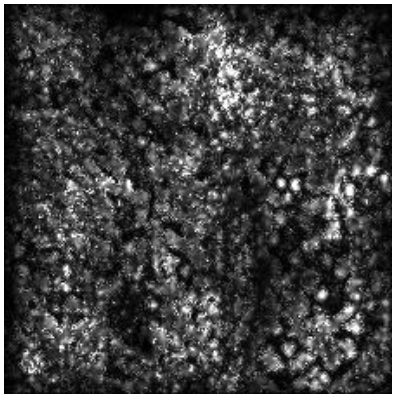
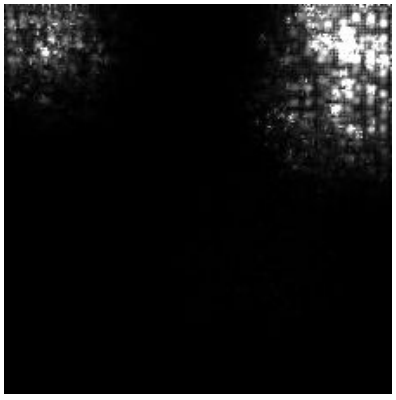
Grad-CAM

The visualizations created by gradcam.py and guided_gradcam.py are used to identify the features of an image the model looks are most to classify it as it does, basically a Class Activation Map for the most important feature map.

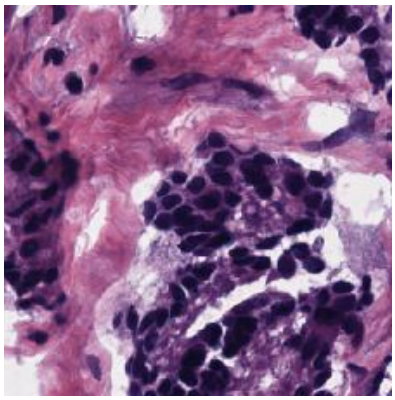
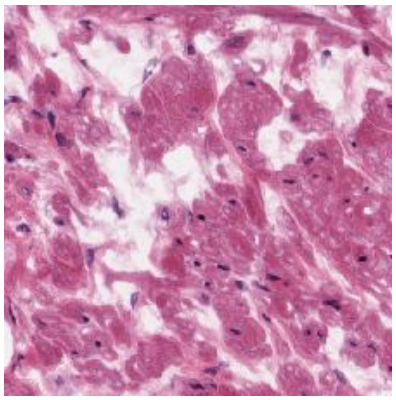
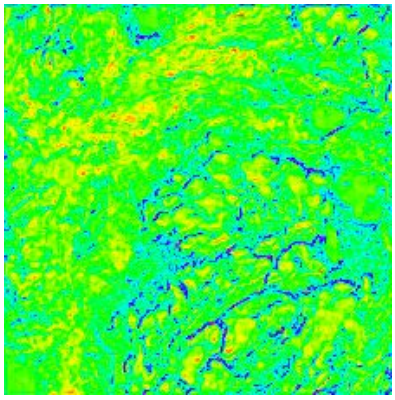
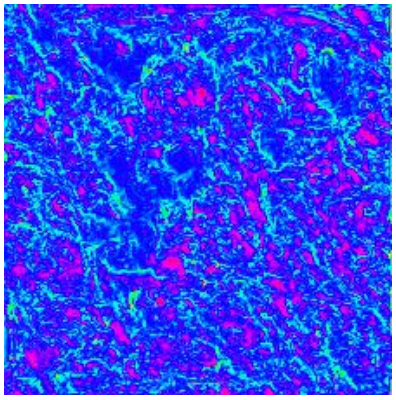
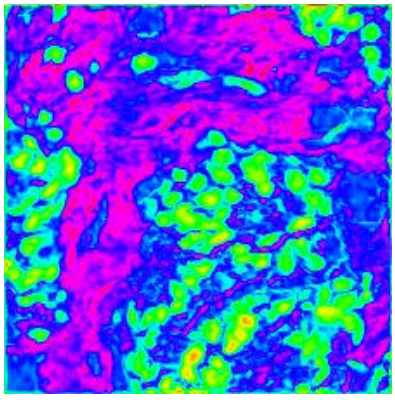
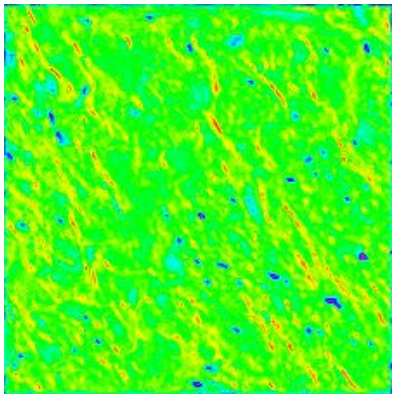
Lung Data

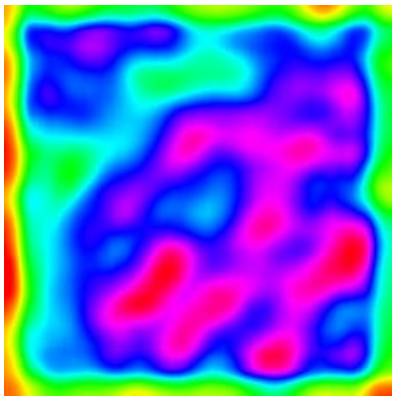
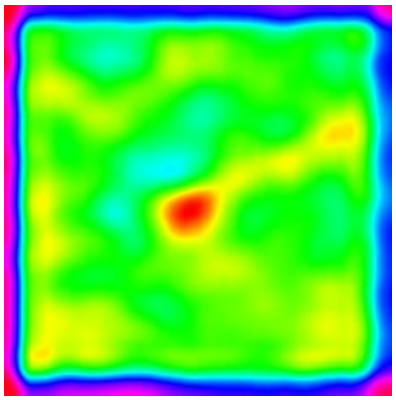
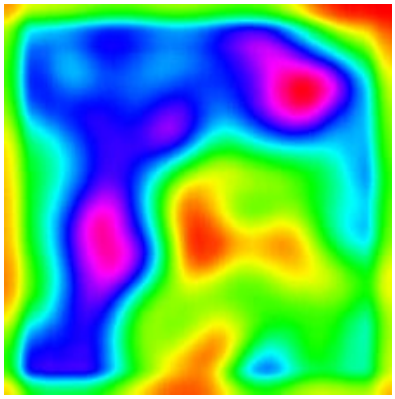
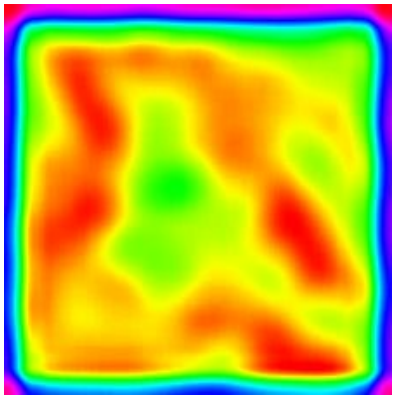
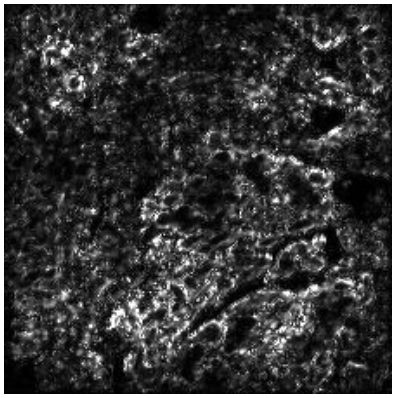
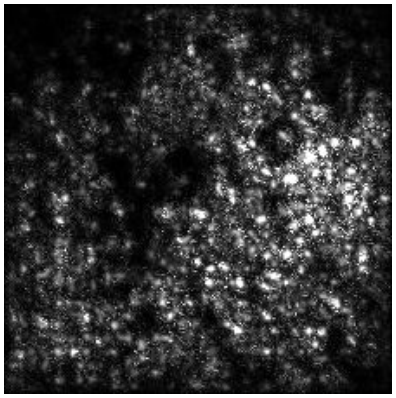
	Cancer	Normal
Original		
Lung Model Grad-CAM Heatmap Block 0 Layer 2		
Prostate Model on Lung Data Grad-CAM Heatmap Block 0 Layer 2		

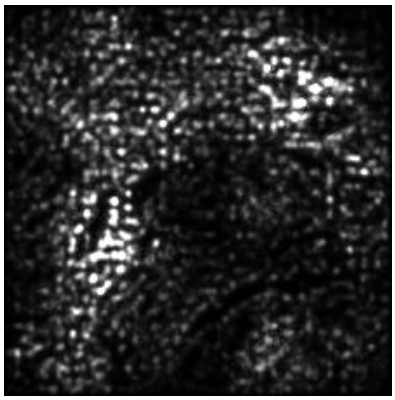
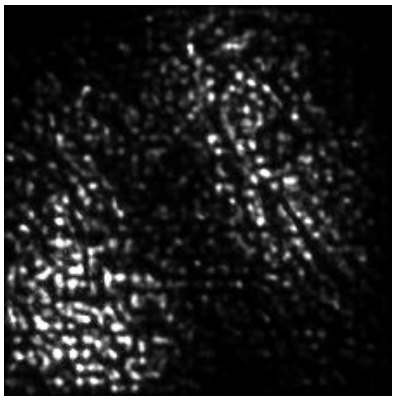
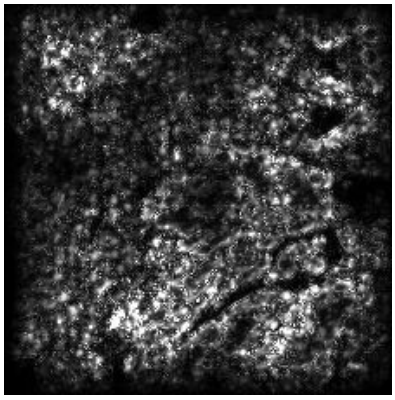
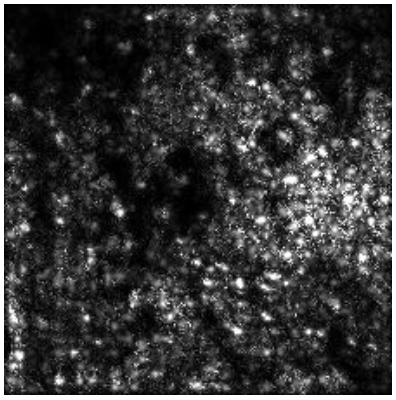
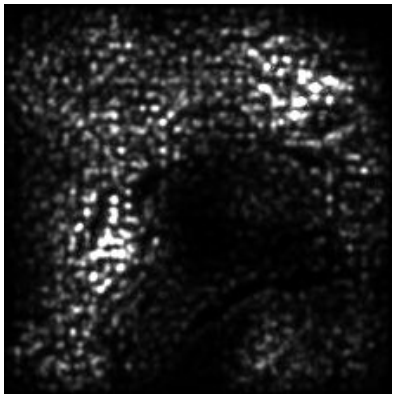
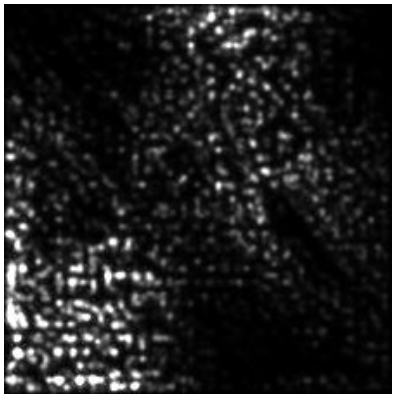
Lung Model Grad-CAM Heatmap Block 2 Layer 14		
Prostate Model on Lung Data Grad-CAM Heatmap Block 2 Layer 14		
Lung Model Guided Grad-CAM Block 0 Layer 2		

Prostate Model on Lung Data Guided Grad-CAM Block 0 Layer 2		
Lung Model Guided Grad-CAM Block 2 Layer 14		
Prostate Model on Lung Data Guided Grad-CAM Block 2 Layer 14		

Prostate Data

	Cancer	Normal
Original		
Prostate Model Grad-CAM Heatmap Block 0 Layer 2		
Lung Model on Prostate Data Grad-CAM Heatmap Block 0 Layer 2		

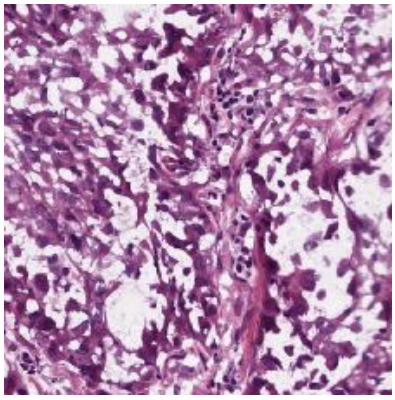
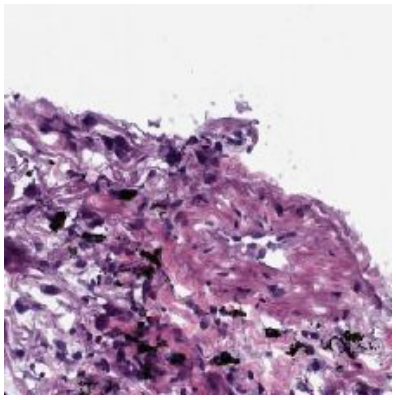
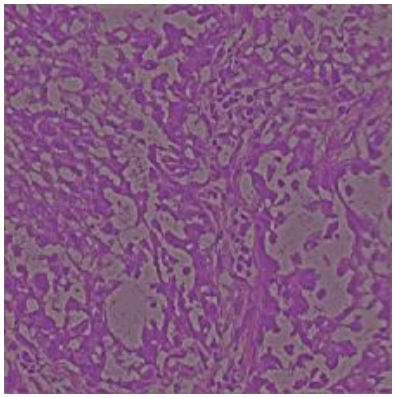
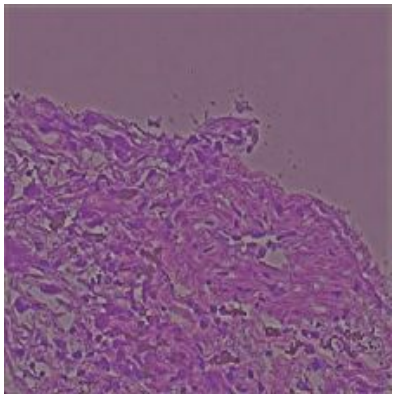
Prostate Model Grad-CAM Heatmap Block 3 Layer 21		
Lung Model on Prostate Data Grad-CAM Heatmap Block 3 Layer 21		
Prostate Model Guided Grad-CAM Block 0 Layer 2		

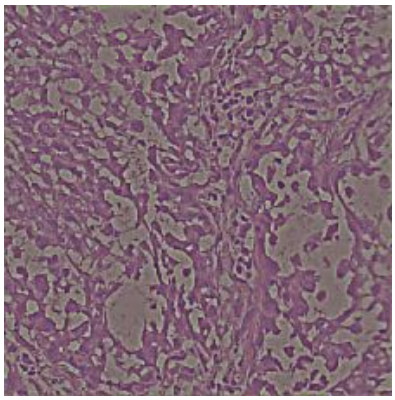
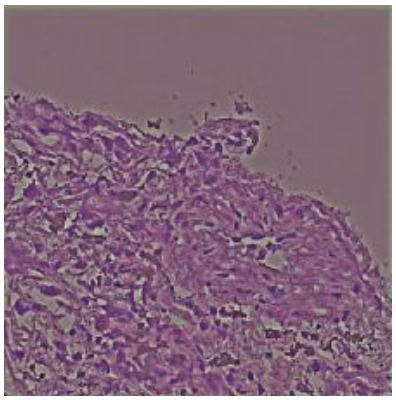
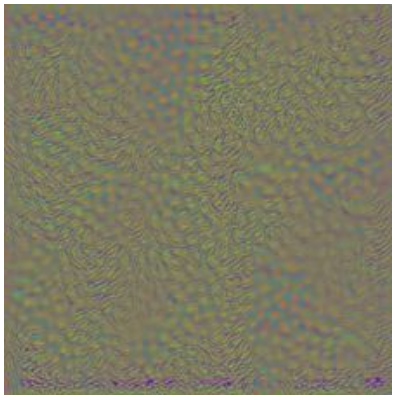
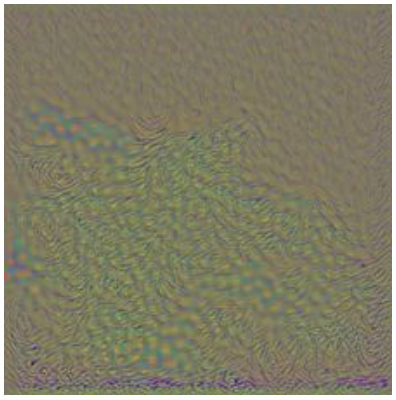
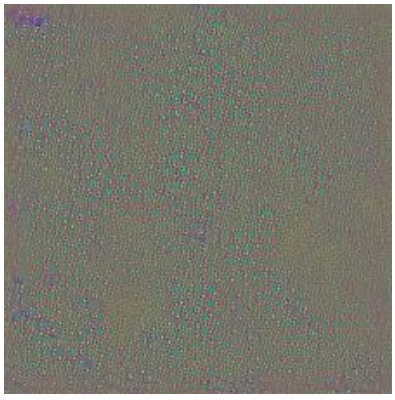
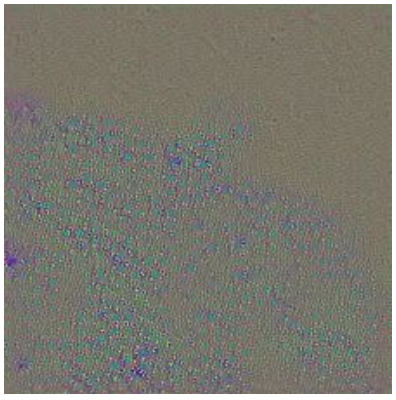
Lung Model on Prostate Data Guided Grad-CAM Block 0 Layer 2		
Prostate Model Guided Grad-CAM Block 3 Layer 21		
Lung Model on Prostate Data Guided Grad-CAM Block 3 Layer 21		

Inverted Representation

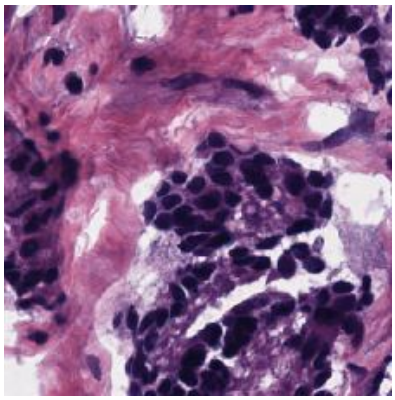
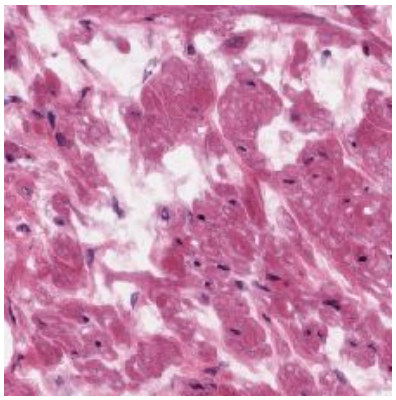
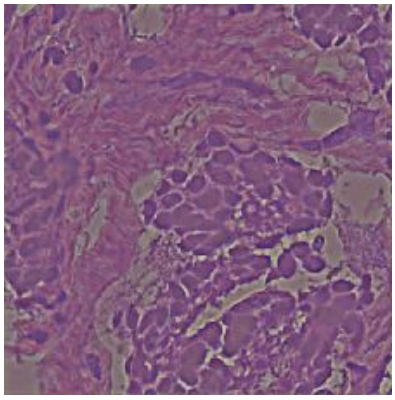
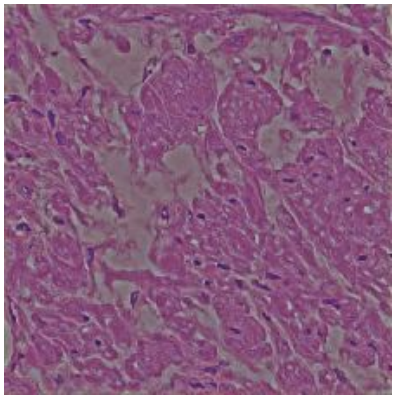
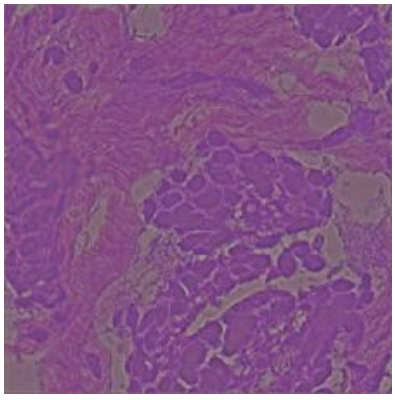
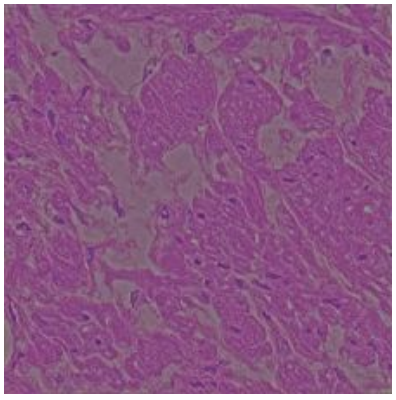
The visualization created by inverted_representation.py uses the result of a filter on an image, and tries to iterate back to the original image. Using this we can find what objects or properties the filter does **not** look at.

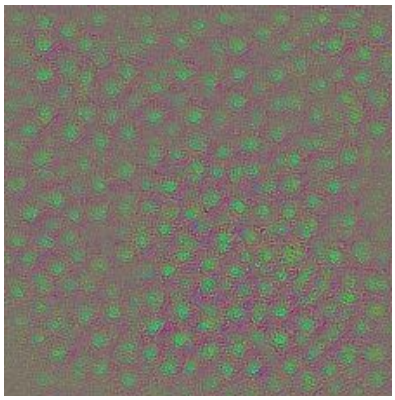
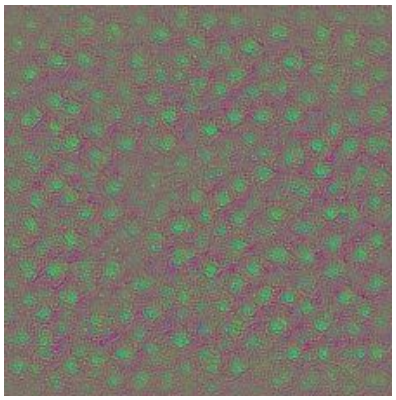
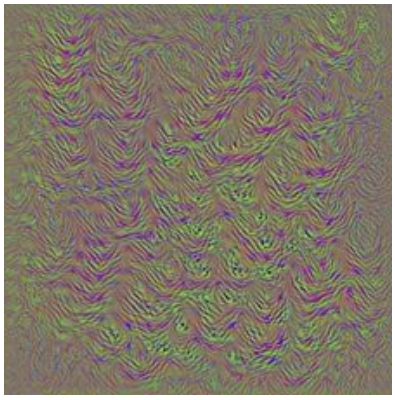
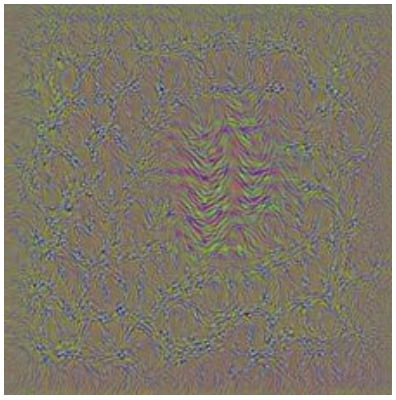
Lung Data

	Cancer	**Normal**
Original		
Lung Model Inverted Representation Block 0 Layer 2		

Prostate Model on Lung Data Inverted Representation Block 0 Layer 2		
Lung Model Inverted Representation Block 2 Layer 14		
Prostate Model on Lung Data Inverted Representation Block 2 Layer 14		

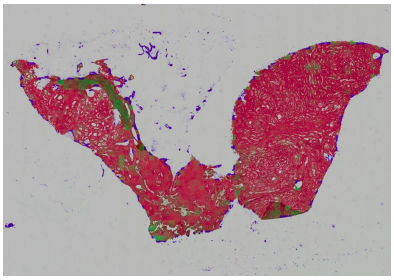
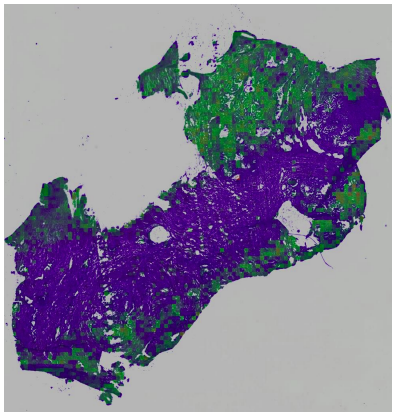
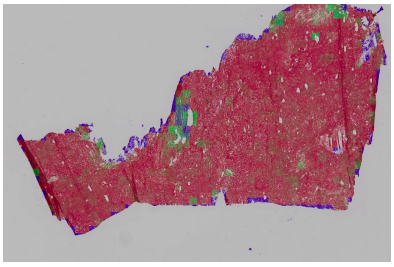
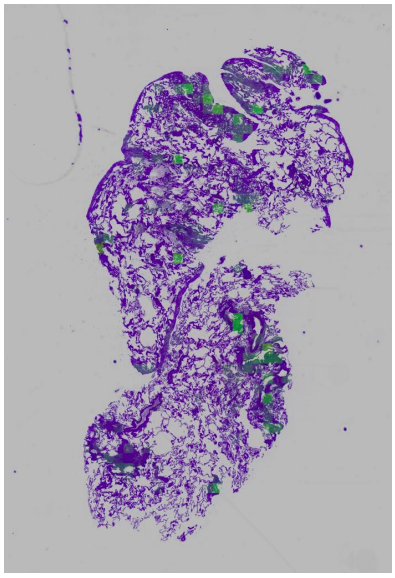
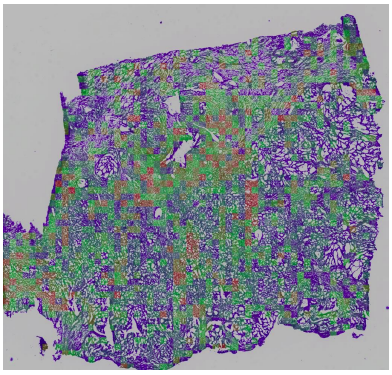
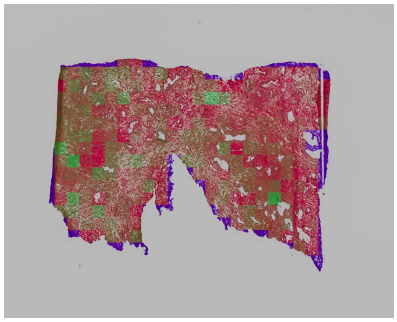
Prostate Data

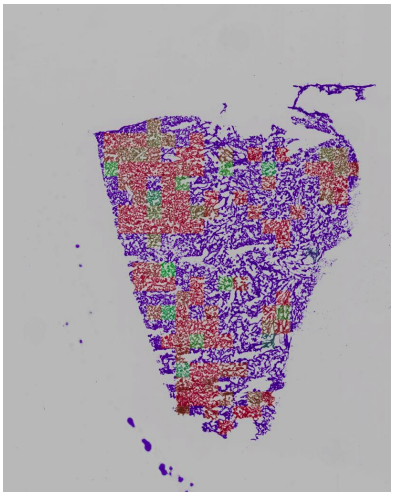
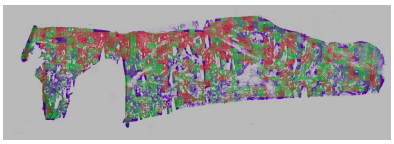
	Cancer	Normal
Original		
Prostate Model Inverted Representation Block 0 Layer 2		
Lung Model on Prostate Data Inverted Representation Block 0 Layer 2		

Prostate Model Inverted Representation Block 3 Layer 21		
Lung Model on Prostate Data Inverted Representation Block 3 Layer 21		

Probability Heatmap

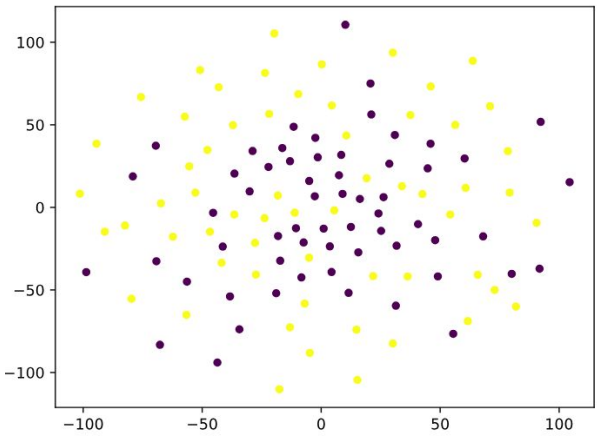
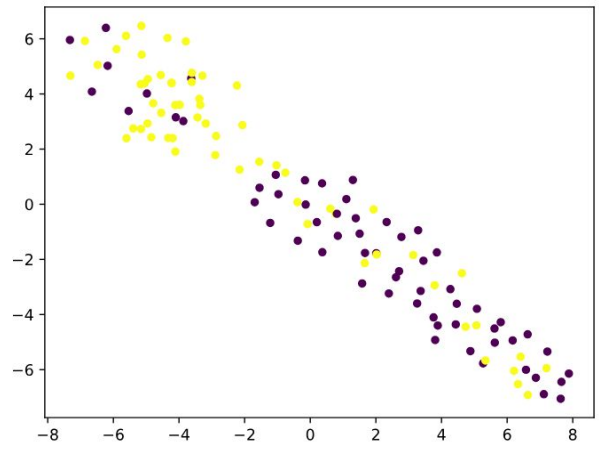
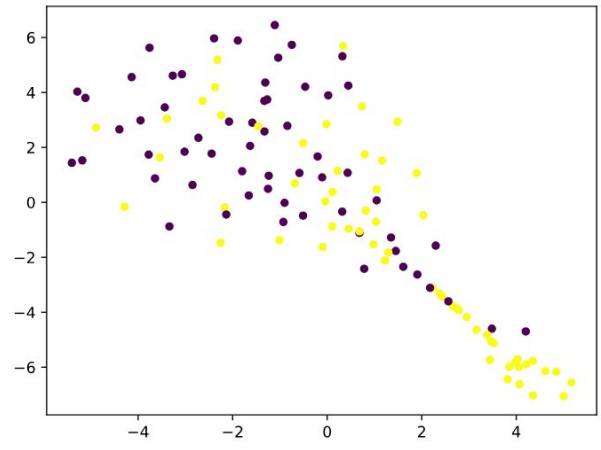
The probability heatmap is created by finding probability of cancer for each patch in the whole dataset using probability_finder.py and the mapping the probabilities to heatmap colours and joining them to find the probability heatmap for all WSIs' using heatmap_generator.py. A few examples are shown below.

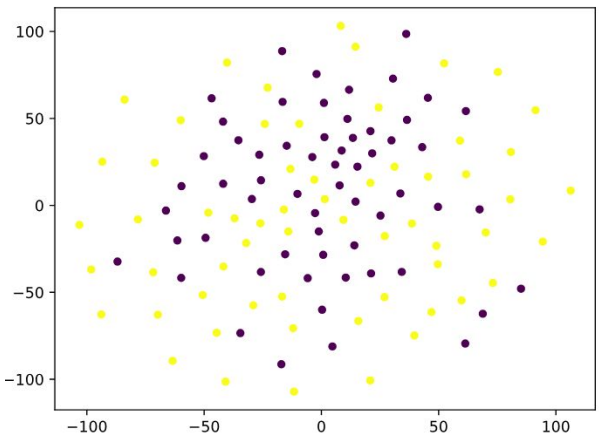
Detection \ Label	Cancer	Normal
Prostate Correct Detection		
Lung Correct Detection		
Prostate Incorrect Detection		

Lung Incorrect Detection		

TSNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The models are visualized based on their seperability of the examples images from both cancer and normal. The 4096 sized feature vector at the end of the model, from the fully connected layer provides us with a high dimensional embedding which is reduced into a 2 dimensional point for each image. A total of 60 patches from each lung and prostate datasets with 20 each from training, validation and test sets have been used to produce the following t-SNE maps shown below.

Model	T-SNE Map
Lung	
Prostate on Lung Data	
Prostate	

Lung on Prostate Data	

Conclusions

The visualizations indicate that fine-tuned models look at more detailed and accurate features that represent the classes.

CODE AND INSTRUCTIONS TO REPRODUCE STEPS

Each step of the process that was followed, from the extraction of data to training and fine-tuning, is explained in order below.

DATA COLLECTION

With the gdc-client and a manifest file that describes all the SVS files needed to be downloaded (as described in the GDC section), downloading of the files can be done using the command. Use https://portal.gdc.cancer.gov/legacy-archive/ as the new gdc repository has omitted some data.

```
gdc-client download -m <manifest file>
```

FILE AND FOLDER STRUCTURE

File Structure

The file name contains some information about the SVS file that includes case number, cancer or normal, and patient ID.

EXAMPLE

TCGA-2A-A8VL-01A-02-TS2.AFBBB2D5-39E6-434A-B6E5-779DD8217DCD.svs

Case ID

Cancer (<=10) / Normal (>10)

Cancer Subtype

Patient ID

Folder Structure

The folder structure that was used to store the data is shown below. Depending on where they are stored, on the data space or temporary memory space of the cosmos node, the structure differs. The data space folder is split into slides and patches from slides, whereas only the patches are loaded into the /tmp/ folder of the cosmos node for faster processing.

DATA SPACE

Cancer Name (Eg. lung)

- **slides**
  - **All .svs files**
- **Patches**
  - **train**
    - **cancer**
    - **normal**
  - **valid**
    - **cancer**
    - **normal**
  - **testi**
    - **cancer**
    - **normal**

PATCH EXTRACTION

For patch extraction, OpenSlide library is required. OpenSlide is actually a C library, but Python provides an interface. Thus, both the C library and the Python interface are to be installed. The patch extractor works in 3 steps:

1. Go through all the slides in <data space directory>/slides and check if they are blank or corrupted by looking at a small portion of the slide using openslide's read_region function

2. Segregate the files into cancer and normal by looking at the file names
3. Load the slides onto a parallel processor and let each process extract patches from one slide. The patches are validated to make sure they have at least 50% tissue content.

```
python patch-extractor.py --help

usage: patch-extractor.py [-h] [-p [{256,512,1024}]] [--equal] data-path

Extracts patches from the highest magnification(level 0) of the svs images

in the data_directory/slides/ containing .svs files only

positional arguments:

 data-path           Path to data directory where slide images are present

optional arguments:

 -h, --help          show this help message and exit

 -p [{256,512,1024}], --patch-size [{256,512,1024}]     Default value is 512. Other choices are 256 and
1024.

 --equal             Equalize data sets

EXAMPLE

python patch-extractor.py /shared/supernova/home/sairam.tabibu/lung
```

The --equal is a relatively new option added to equalize the number of slides and patches obtained from cancer and normal WSIs. It finds out a threshold minimum patches that a slide should provide in order to be considered. If there are 710 slides in total, with 594 cancer and 116 normal, the normal slides' dimensions are taken into account to find out

the number of patches each one can give. Based on this information a suitable limit is set that gives the maximum number of patches in total with equal participation from cancer and normal slides.

TRAINING

The training is done using a VGG-16 model from pytorch's models module loading data form the image folder in /tmp/ using pytorch DataLoader into datasets for training and validation. Most of the hyper-parameters are modifiable from the command line arguments. The training also calculates metrics like accuracy, precision, recall and f2-score.

```
usage: fscore_vgg.py [-h] [-s [SIZE]] [-l [LR]] [-b [BATCH]] [-w [WEIDECAY]] [-d [DROPOUT]]
[-bo [BETAONE]] [-bt [BETATWO]] [-e [EPOCHS]] [-f [FINETUNE]] [-n [NUMFREEZE]] [--batchnorm]
[-r [RATIO] name data-dir log-dir

Constructs a pretrained imagenet model and fine-tunes it with the dataset provided. It also stores
logs and models of the training in given log_dir.

positional arguments:

 name                Name of dataset. Will be used as suffix for data and log directories

 data-dir            Path to data directory where images are present in folders train, valid and testi,
under respective classes cancer and normal

 log-dir             Path to log directory where logs and final models will be stored

optional arguments:

 -h, --help          show this help message and exit

 -s [SIZE], --size [SIZE] Image size to be transformed into. Default value is 224

 -l [LR], --lr [LR]    Learning rate. Default value is 1e-4
```

```
-b [BATCH], --batch [BATCH]   Batch Size. Default value is 128

-w [WEIDECAY], --weidecay [WEIDECAY] Weight Decay. Default value is 0

-d [DROPOUT], --dropout [DROPOUT]  Dropout rates. Default value is "0.5,0.5"

-bo [BETAONE], --betaone [BETAONE]   Beta 1. Default value is 0.9

-bt [BETATWO], --betatwo [BETATWO]   Beta 2. Default value is 0.999

-e [EPOCHS], --epochs [EPOCHS]  Epochs. Default value is 15

-f [FINETUNE], --finetune [FINETUNE]  Fine-tune from. Default from ImageNet.

-n [NUMFREEZE], --numfreeze [NUMFREEZE]   Freeze until nth layer. Default None.

--batchnorm          Batch Normalization

-r [RATIO], --ratio [RATIO]  Cancer:Normal ratio. Default 1.
```

EXAMPLE

python fscore_vgg.py prostate /tmp/ /shared/supernova/home/sairam.tabibu/training/ -b 96 -l 7e-5 -d "0.5,0.5" -w 2e-3 -r 4.93333333 --finetune /shared/supernova/home/sairam.tabibu/results/lung/ -n 2

Train prostate model with data from /tmp/prostate/ with weights loaded from the stored lung model, and log the progress in training folder. Use batch size of 96, 7e-5 learning rate, 0.5 dropout, 2e-3 weight decay, ratio of cancer:normal :: 4.9333333:1 and freeze until layer 2.

Training saves different files for different metrics for both validation and testing separately in log_dir. Training also stores model state for best F2 and error. Training can be resumed from last epoch if interrupted, using the model state stored for the best loss or f2 so until that epoch.

TESTING

The testing phase is very similar to the training phase, with fewer options..

usage: vgg_tester.py [-h] [-s [SIZE]] [-p [{testi,train,valid}]] [--batchnorm] model-name test-name data-dir log-dir

Tests existing model against a test dataset. It also stores logs of the test in the given log_dir.

positional arguments:

model-name Name of the dataset on which model was trained. Will be used as a suffix for data and log directories.

test-name Names of datasets, comma separated without blank spaces. Will be used as a suffix for the log directory where the model should be.

data-dir Path to data directory where dataset folders are present.

log-dir Path to log directory where logs and models for datasets are to be found.

optional arguments:

-h, --help show this help message and exit

-s [SIZE], --size [SIZE] Image size to be transformed into. The default value is 224

-p [{testi,train,valid}], --phase [{testi,train,valid}]

```
--batchnorm           Batch Normalization
```

EXAMPLE

```
python vgg_tester.py lung lung /tmp/ /shared/supernova/home/sairam.tabibu/results/

Lung model from results tested on lung data in /tmp/
```

VISUALIZATIONS

Visualizations are stored as image files as well as shown on a visdom server that should be live on running the visualization programs. A common structure for all all the visualization programs is given below. The code is well commented and the explainations are also provided on pytorch-cnn-visualizations in github.

```
usage: <vizprogram> [-h] [--batchnorm] [-s [SERVER]] [-t [TARGET]] [-l [LAYER]] model-dir

Visualize <vizprogram> using Visdom.

positional arguments:

 model-dir            Directory where model for dataset is to be found.

optional arguments:

 -h, --help           show this help message and exit

 --batchnorm          Whether Batch Normalization was used.

 -s [SERVER], --server [SERVER] Server address. Default http://10.4.16.22 (i.e. node13).

 -t [TARGET], --target [TARGET]  Target number. Default 0.
```

```
-l [LAYER], --layer [LAYER] Layer number. Default 10.

-f [FILTER], --filter [FILTER] Filter number. Default 5.
```

Source code, results and visualizations are available at Adenocarcinoma-Transfer-Learning Github private repository, only viewable by collaborators. The create-mini-dataset.py is for creating smaller datasets preserving the same ratio between cancer and normal.