Georgia Institute of Technology

For AI+ First Technology Co., Ltd.

Team 3: Final Report

Melirose Liwag, Mehmet Gumus, Charles Daniels

7-19-2023

Contents

1 Team Introductions

Charles Daniels graduated in 2016 with undergraduate degrees in chemical engineering and chemistry from the University of Michigan; he stayed at the same institution and earned a master's degree in energy systems engineering before working at General Motors. There, he created physical simulations of engine operation to optimize fuel economy and power output. He currently works in Boston as a process controls engineer with hydrogen fuel cells. Charles is excited to apply his learnings to such an interesting project.

Melirose Liwag graduated in 2020 with a Bachelor of Science in Digital Arts Engineering from the University of Silicon Valley (Formally known as Cogswell Polytechnical College). There, she was first introduced to the programming field and fell in love. To further her curiosities, she decided to broaden her view on programming languages (mainly Python) which brought her to the OMSA program at Georgia Institute of Technology. Despite being new to the field, Melirose has a keen eye for learning and is excited to end the program strong with a great project.

Mehmet Gumus is a teaching assistant professor in the Department of Mathematics and Statistics at the University of Nevada, Reno. He received his Ph.D. in Mathematics from Southern Illinois University Carbondale in 2017. Prior to joining the University of Nevada, Reno in 2019, Dr. Gumus worked as a Visiting Assistant Professor in the Department of Mathematics and Statistics at Auburn University. His primary research interests are in linear algebra, matrix computations, and machine learning. Specifically, he is interested in high dimensional data analytics and image processing.

2 Project Background

Female breast cancer has currently surpassed lung cancer as the most commonly diagnosed cancer and is now the fifth leading cause of death for women globally (Cancer.Net Editorial Board, 2023). Ultrasound is a useful tool in the diagnosis of breast cancers: it is used to help differentiate cysts from solid nodules, evaluate palpable lesions, and is helpful in case of a patient history involving lumpectomy or segmentectomy (Devolli-Disha, Manxhuka-Kërliu, Ymeri, & Kutllovci, 2009). Ultrasound also does not utilize damaging radiation, is more cost effective than other imaging techniques, uses conventional power infrastructure, and provides real-time imaging (Brattain, Telfer, Dhyani, Grajo, & Samir, 2018). There is also evidence that ultrasonography is more accurate for younger women (under the age of 45) when compared against mammography (Devolli-Disha, Manxhuka-Kërliu, Ymeri, & Kutllovci, 2009). Computer-aided diagnosis (also known as CAD) has been previously deployed to help radiologists in classifying breast tumors. In the past, statistical methods have been used to extract and classify tumors based on lesion shapes, margins, homogeneity, and posterior acoustic attenuation (Giger, Al-Hallaq, Huo, Wolverton, & Chan, 1999). This traditional approach, however, can be difficult when ultrasound images vary so widely and the successful extraction of features is completely dependent the machine and the operator of said ultrasound machine.

As an example of this, Wei et al (Wei, et al., 2020) explored classification of texture and morphological features of Ultrasound images in a recent project in 2020. The team utilized these features to further improve the accuracy of benign and malignant tumor classification, resulting in an average accuracy of 91%. This result was obtained by building and comparing models on the principal components of each feature. Not only did this technique increase classification accuracy, but utilizing PCA also increases the training speed of the algorithm due to the high dimensionality of the textural features. Despite this, the method described in this project may not be well implemented with the use of single (or multiple) classifiers considering the differing preprocessing stages required for both CNNs and classifiers mentioned previously. It is interesting to note, however, that other classifiers have been used with some success in this space.

Newer deep learning algorithms, on the other hand, show potential for less rigidity in the classification of ultrasound images. Deep learning approaches can extract non-linear features from data and are therefore particularly useful in situations where complex pattern recognition is needed (Brattain, Telfer, Dhyani, Grajo, & Samir, 2018). This avoids the fragility of traditional statistical methods but may require more data to train successfully. Convolutional Neural Networks (CNNs) are of particular interest because of their ability to grapple with image-based tasks.

A large diversity of deep learning approaches has been applied to ultrasound data of breast lesions. In 2017, a team of researchers trained a GoogLeNet CNN with ultrasound images to differentiate between benign and malignant tumors in breast lesions (Han, et al., 2017). Said CNN had a final accuracy of about 90%. Other researchers have utilized deep-learning techniques to help classify breast tumors with shear-wave elastography (a form of ultrasound that can help characterize tissue hardness); their research focused on training their networks to discriminate between task-relevant and task-irrelevant data (automated “feature-selection”) on shear-wave elastography data in particular (Zhang, et al., 2016). Attention-gating mechanisms have also been used in conjunction with traditional CNNs to help suppress noise and highlight relevant features in ultrasound images with breast tissue lesions (Kalafi, et al., 2021).

A deep-learning CNN algorithm, AlexNet, was used by Yi et al (Yi, et al., 2017) in a 2017 research project exploring the classification of benign and malignant tumors from Ultrasound images provided by the publicly available Digital Database for Screening Mammography (DDSM). Two methods of training were explored (specifically Parallel training and Multi-Modal training) for images with differing viewpoints of the data. Initially built with the AlexNet algorithm, the team later concluded that a model built with GoogLeNet produced the best results with an accuracy of 0.84 and an AUC of 0.85. Deep-learning algorithms with segmentation were explored by Rouhi et. al (Rouhi, Jafari, Kasaei, & Keshavarzian, 2015) and further improved classification accuracy to about 96%.

In this project, we evaluated three popular CNN architectures before selecting one to focus on; our goal is to have the CNNs successfully classify as many benign and malignant tumors as possible. After that, we discuss our efforts in object detection as well.

3 Methods & Model Architecture

Currently, the initial phase of this project is focused around using three basic CNN architectures to help classify ultrasound images of benign and malignant lesions. The results of this phase will help establish a "baseline" accuracy for each of these architectures and will help guide future steps for classification improvement. The goal of this project is to help doctors analyze ultrasounds more efficiently and more precisely, so maximum classification accuracy is the desired outcome of this work. The three basic CNN architectures we will explore initially are:

1. ResNet18
2. DenseNet
3. MobileNet

For the object detection portion of this work, we utilized the "You Only Look Once" or YOLO model.

3.1 ResNet

3.1.1 Background

A CNN (Convolutional Neural Network) is a type of deep learning algorithm designed for image classification problems. Research shows that network depth is key for classifying images with high accuracy (Huang, Liu, Van Der Maaten, & Weinberger, 2017). However, deeper networks lead to a degradation in performance because additional layers cause higher training errors. A deep residual learning framework (ResNet) addresses this problem by adding identity mapping as a new layer (see Figure 1) (He, Zhang, Ren, & Sun, 2016). This method allows for the construction of a deeper network with no loss in training error compared to a plain architecture. This process doesn't increase the computational complexity while keeping the number of parameters same.

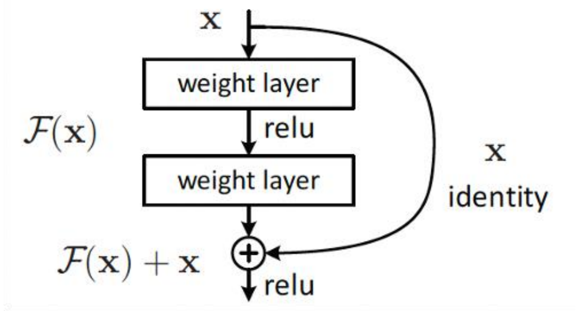
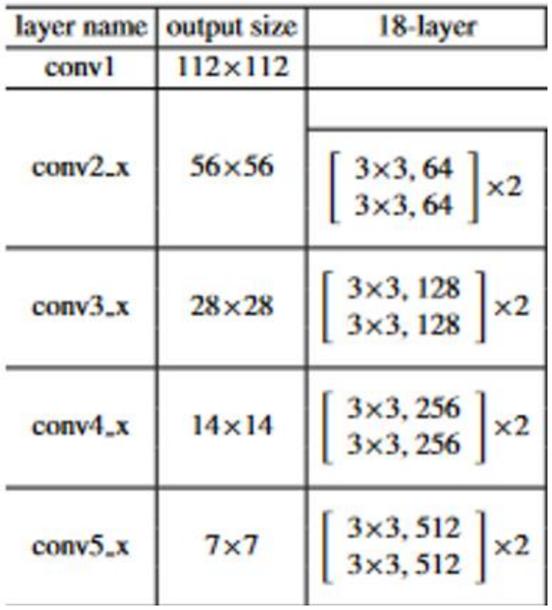

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 1. Residual learning: a building block (He, Zhang, Ren, & Sun, 2016) (He, Zhang, Ren, & Sun, 2016)

If x and F(x) in Figure 1 do not have the same dimension, the identity mapping is multiplied by a linear projection W so that x and F(x) can be combined as input to the next layer. The formulation of F(x)+x can be obtained by feedforward neural networks with shortcut connections, which skip one or more layers. This extension to CNN increases the performance in image classification (He, Zhang, Ren, & Sun, 2016).

3.1.2 ResNet18 Model and Architecture

The ResNet-18 architecture used in our project has 18 trainable layers including 1 input layer, 8 convolutional layers, 4 sets of residual blocks, and 1 fully connected layer. The input layer takes an image of a set size and outputs probabilities that the given image belongs to a particular class. The base architecture is referenced from He et al. and the details of the architecture can be seen in Figure 2 below. The baseline uses a regular ReLU activation function but ReLU6 will be explored in this report.



layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$

Figure 2: ResNet Architecture (He et al.)

The initial convolutional layer is with 7x7 kernel and a stride of 2, followed by batch normalization and ReLU activation. This layer reduces the spatial dimensions of the input by half. After the first convolutional layer, a max pooling layer with a 3x3 kernel, a stride of 2 and a padding of 1 is applied. It further reduces the spatial dimensions while increasing the depth of the feature maps. After the input layer, there are four sets of residual blocks, which include two 3x3 convolutional layers, each followed by batch normalization and ReLU6 activation. These blocks support skip connections. Identity down sampling is employed by skip connections to ensure the matching of input and output dimensions. The input is added to the output of the second convolutional layer to form the residual connection. The inclusion of skip connections through the residual blocks mitigates the issue of vanishing gradients, thereby facilitating the training of deep neural networks. After the last set of residual blocks, a global average pooling layer is applied. It computes the average value for each feature map, resulting in a fixed-length vector regardless of the input size. A fully connected layer is used to produce the final classification output.

3.2 DenseNet

3.2.1 Background

The second CNN, DenseNet (Huang, Liu, Van Der Maaten, & Weinberger, 2017), seeks to solve several problems that plague increasingly deep CNN architectures: the vanishing-gradient problem (where information on the loss-function gradually erodes as it passes through many layers, effectively not training earlier layers in deep networks), the inability of important features to directly propagate throughout the CNN, and the ballooning of the number of parameters that need to be trained in large CNNs. These problems are solved through the simple idea that all previous layer outputs should be forwarded to each successive layer's input – this guarantees that each successive layer in the CNN "sees" the output of each previous layer directly.

With this change, the loss-function information more easily reaches early layers (alleviating the vanishing-gradient problem and helping ensure earlier layers are trained effectively). Features in earlier layers of the CNN are also fed directly to other layers in the CNN rather than being successively filtered by chained layers (eliminating redundancy in deep networks). Lastly – and perhaps surprisingly - the number of total parameters that need to be trained in DenseNets decrease compared to the number of total parameters required in comparable ResNets and VGG networks. It is important to note that ResNets and Highway Networks have hinted that this kind of deeply interconnected approach is effective by showing that skip connections and bypass paths are beneficial to CNN performance.

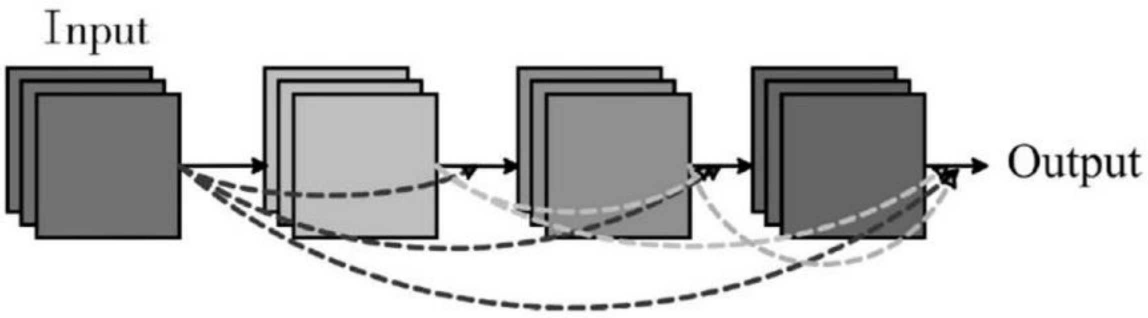
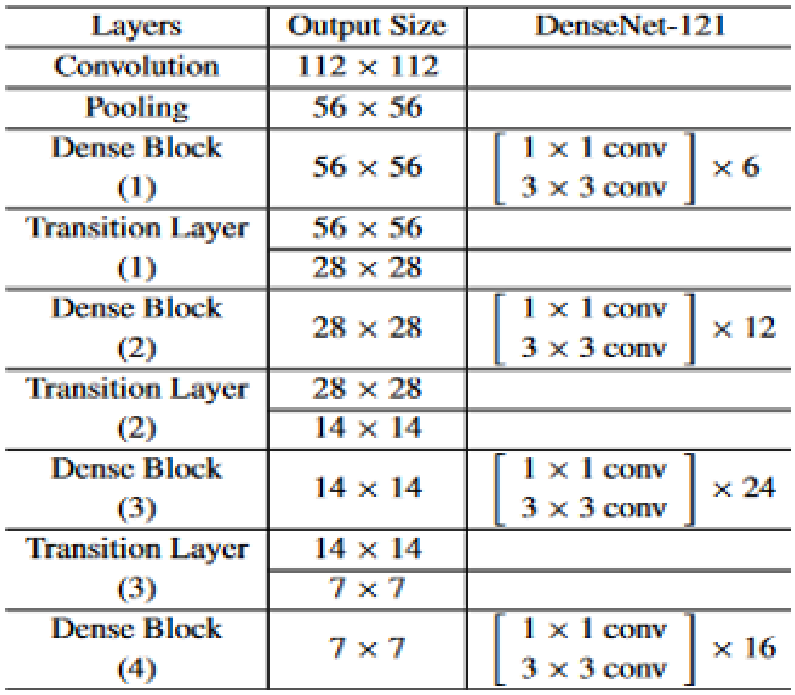


Figure 3: Example of a "dense block" within a dense neural network, where layer outputs are linked to the input of each successive layer (Zhang, Wu, Yu, & Lei, 2021)

3.2.2 DenseNet 121 Model and Architecture

The base architecture of DenseNet 121 is built in Python using PyTorch and is referenced from Zhang et al. with the standard kernel size 7, and 64 initial features for the baseline model. These hyperparameters, and others, can be adjusted and experimented to achieve reasonable accuracy. A review of this model's performance will be shown later in Table 2 and Table 3.



Layers	Output Size	DenseNet-121
Convolution	112×112	
Pooling	56×56	
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	
	28×28	
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	
	14×14	
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14×14	
	7×7	
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$

Figure 4: DenseNet 121 Architecture (Zhang et al.)

3.3 MobileNet

3.3.1 Background

Lastly, our team decided on comparing the intricacies of the MobileNet models, specifically with MobileNet v1 and MobileNet v2. The MobileNet models aim to improve the speed and latency of CNNs and while still providing reasonable accuracy comparable to other network designs. By using depthwise separable convolution layers, a MobileNet model is 8 to 9 times faster (but with less accuracy) when compared to the speeds of other comparable networks (Howard, et al.). This is achieved by filtering inputs through a depthwise convolution filter as shown in Figure 5.

Depthwise convolution filters seek to relieve the computational burden of running standard convolutional layers; this key optimization allows MobileNet to make more effective trade-offs in speed and accuracy.

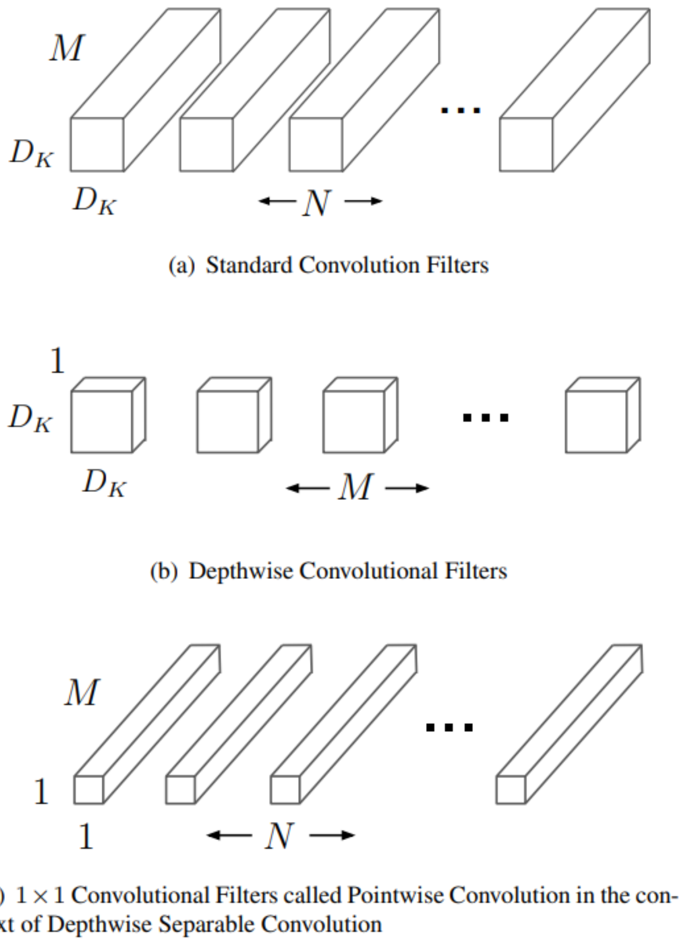
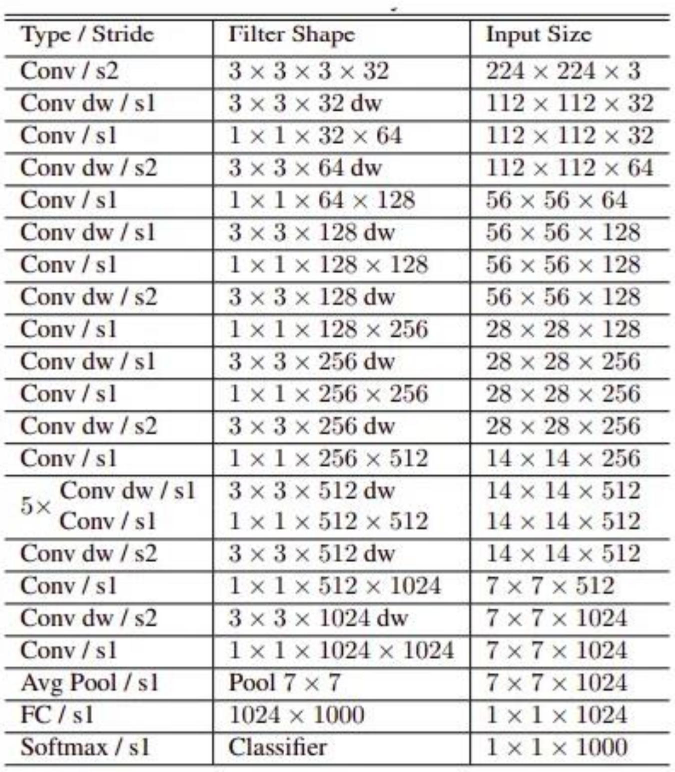

M
D_K
D_K
$\leftarrow N \rightarrow$
(a) Standard Convolution Filters
1
D_K
D_K
$\leftarrow M \rightarrow$
(b) Depthwise Convolutional Filters
M
1
1
$\leftarrow N \rightarrow$
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 5: The standard convolutional filters in (a) are replaced by two layers: a depthwise convolution layer (b) and pointwise convolution layer (c) to build a depthwise separable filter (Howard, et al.)

Additional layers are required after the depthwise convolution filter layers shown above as these layers alone do not output new features and only act upon input channels. The outputs of the depthwise convolution layer are modified by a series of 1 X 1 Convolution filters, which then build the final depthwise separable convolution layer as described by MobileNet. This process of factorization reduces the computation times drastically compared those of a standard convolution filter.

3.3.2 MobileNet v1 Model and Architecture

The standard architecture of MobileNet v1 contains 3 X 3 Depthwise Separable Convolution layers with the addition of batchnorm and ReLU nonlinearity functions in between each layer, with the final structure of MobileNet v1 standing at about 28 layers. From Figure 6, we built MobileNet v1 in Python following the architecture provided by Howard et al. Building a base architecture of the model will help give us a baseline metric when finding hyperparameters that fits the data well. The base structure takes in an input of 224x224 with kernel size 3x3.

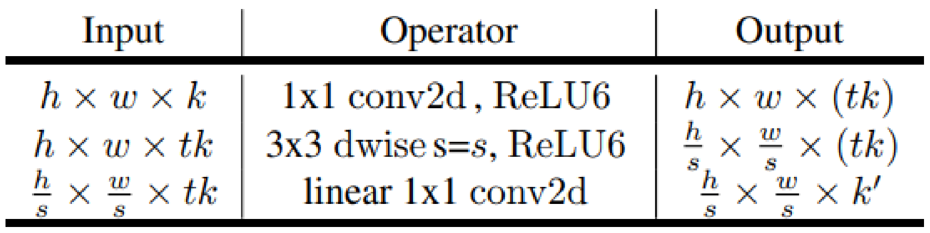


Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 6: MobileNet v1 architecture (Howard et al.)

3.3.3 MobileNet v2 Model and Architecture

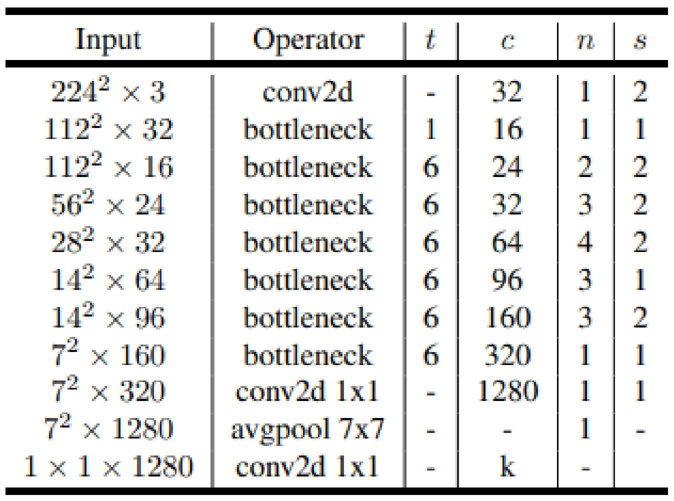
Similar to MobileNet v1, MobileNet v2 utilizes the depthwise separable convolution structure but also uses a width multiplier and several linear bottleneck layers to further reduce computation time. MobileNet v2 exploits the idea of dimensionality reduction of a layer which would reduce the dimensionality of the operating space. This reduction in dimensionality would affect the accuracy of the model but also greatly affect the computation time. To achieve this, Sandler et al. added a width multiplier hyperparameter to assist in dimensionality reduction. The team found, through supplemental material, that the ReLU function could preserve all the information contained in the input if the input lies in a “low-dimensional subspace of the input space“ (Sandler, et al.) as information could be lost (only in that channel) when the channel is collapsed. This led to the addition of Linear Bottleneck Layers which was found to be crucial in preventing non-linearities, such as ReLU, from destroying too much information.



Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Figure 7: Bottleneck depth-separable Convolution Layer with Inverted Residuals block. Transforms from k to k' channels, with stride s, and expansion factor t. (Sandler, et al.)

To connect the bottleneck layers, inverted residual blocks were used as a shortcut instead of regular residual blocks. This inverted design of the residuals also helped in making the model memory efficient. An example of a single bottleneck layer block is detailed in Figure 7 above. The resulting MobileNet v2 model is an architecture of 32 filters, 19 residual bottleneck layers, and 3.4 million parameters with the standard width multiplier of 1 and input resolution of 224 x 224. Similar as before, the base architecture for MobileNet v2 is gathered from Sandler et al. Further insights on how the MobileNet models fair against the other models we explored will be detailed in a later section of this report.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 8: MobileNet v2 Architecture as described by Sandler et al.

3.3.4 MobileNet v1 vs. MobileNet v2

Additionally, we can compare both base structures of the MobileNet models on the metrics of accuracy and computation time. As shown previously, MobileNet v2 has fewer parameters than MobileNet v1, namely 3.4 million versus 4.2 million parameters respectively. The bottleneck approach greatly reduced the number of parameters and should reduce computation time, as we have seen a big difference in runtime by comparing MobileNet v1 against ResNet and DenseNet alone. However, it is still crucial to see a reasonable output from both models despite the reduction in computation time.

Model	Accuracy (30 Epochs)	Runtime (50 Epochs w/ GPU usage)
MobileNet v1	79%	~ 2.5 minutes
MobileNet v2	85%	~ 5 minutes

Table 1: MobileNet v1 and MobileNet v2 comparison on accuracy and computation time. Computation time is measured after the model architecture is built and stopped right after the last epoch ends.

Keeping in mind that these models are still baseline models, it is interesting to note that MobileNet v2 achieved a higher accuracy than its predecessor with accuracies 85% and 79% respectively at the end of 30 epochs. MobileNet v1 achieved the highest accuracy of 82% during the 30 epochs whereas MobileNet v2 achieved the highest accuracy of 91% during the 30 epochs. The preservation of information within MobileNet v2 is prevalent as there is a clear difference in performance between MobileNet v1 and MobileNet v2. It is also important to note that MobileNet v2 outputs a 4-dimensional tensor as opposed to a 2-dimensional tensor (as did ResNet, DenseNet, and MobileNet v1) and thus had to be reshaped to compensate for the structure of our team's code. This reshaping may be the cause of slower computation time and contradicts the theory mentioned above that MobileNet v2 is computationally faster than MobileNet v1. Additionally, the accuracy of MobileNet v2 is almost unreliable due to this reshaping.

3.3.5 MobileNet v2 Discussion

Our experiments with MobileNet v2 successfully improved the accuracy from MobileNet v1 however, the slower computation time and unreliable accuracy did not convince our team to proceed with MobileNet v2. As mentioned before, the trade-off between computation time and accuracy will be crucial especially in classification between benign or malignant tumors. This decision will be clear in later sections discussing the performance of our baseline models.

3.3.6 YOLOv5 Discussion

For object detection, we utilized the "You Only Look Once" algorithm (or "YOLO") to help identify the bounding boxes that correctly frame the tumors inside the image dataset (Redmon, Divvala, Girshick, & Farhadi, 2016). This method was novel because object detection at the time relied on classifiers to handle object detection; instead, YOLO presents the problem as a regression problem. YOLO comprises of a single neural network that predicts both the bounding boxes and class probabilities, and it works on entire images. It is also known for its quick run-time, so like MobileNet it can be used on image streams or videos as well.

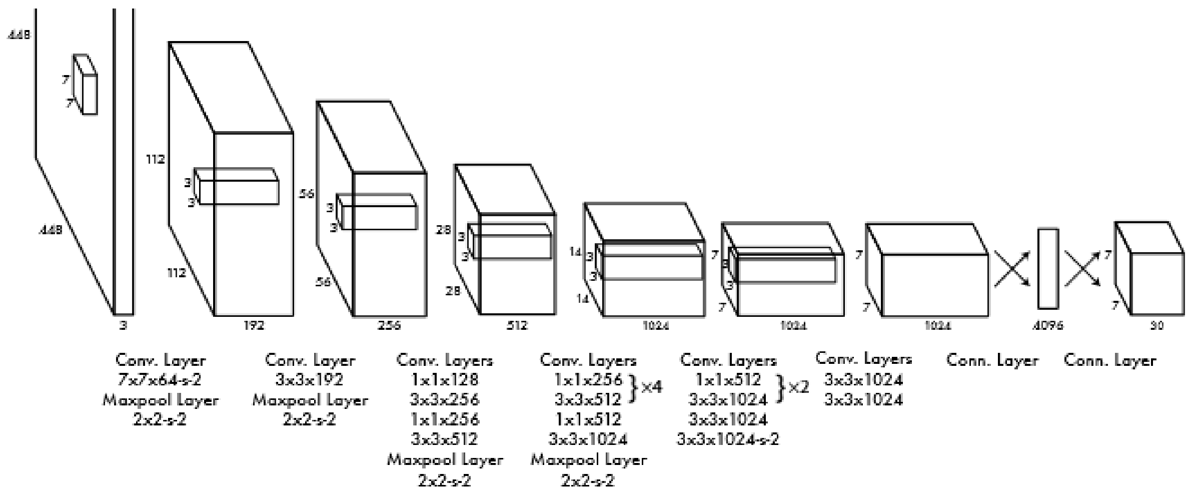


Figure 9: Network architecture for initial version of YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016)

The YOLO concept is built straightforwardly – a single CNN will output simultaneous predictions for the bounding box and class probabilities for whatever is currently being considered. It is a one-stop-shop for object detection and classification, and its CNN architecture is built with that goal in mind.

There are multiple versions of YOLO currently available. For this project we utilized YOLOv5 (an older version that was built entirely on top of PyTorch), as we found it easiest to use with our code base. It's important to note that YOLOv5 can classify images as well as perform object detection, and we explored that capability in our project – as it turns out, the YOLOv5 architecture both performed well on object detection and with tumor classification. This ability to do both the object detection and classification task at the same time makes it an attractive option for this work.

4 Results and Discussion

Our initial results indicated that ResNet18 gave the most robust results in terms of accuracy, precision, recall, and F1 score on both the training and validation dataset. Because of constraints on training time, we decided to focus our optimization efforts on the ResNet-18 model.

We also utilized the YOLOv5 model to help perform object detection and also evaluated its proficiency at classifying each tumor. YOLOv5 proved to be the most accurate model that we tried, and we would recommend its implementation in this application.

4.1 Baseline Models: Results Summary

The following summary is using the preliminary data we acquired from AI+ First with just the baseline models midway through the project. As shown in Table 2 and Table 3, ResNet had the best overall scores on both the training and validation dataset (training set is split into training and validation as the official test dataset was not used yet). These results were obtained after 50 epochs while using the 'Cross-Entropy' loss function and the 'Adam' optimizer.

	ResNet18	MobileNet v1	DenseNet 121
Accuracy	100%	92%	92%
Precision	100%	90%	99%
Recall	100%	92%	83%
F1 Score	100%	91%	90%

Table 2: Classification using Training dataset from initial training data

	ResNet18	MobileNet v1	DenseNet 121
Accuracy	90%	84%	86%
Precision	90%	80%	92%
Recall	90%	80%	70%
F1 Score	90%	80%	79%

Table 3: Classification using Test dataset from initial training data

The results above show that ResNet18 gives the best results on most metrics on both the training and validation datasets, but the difference between the training dataset metrics and the validation dataset metrics indicate that overfitting is a concern for all model types.

Going forward from this point on, we chose to focus on improving the ResNet18 model due to time constraints. More specifically, our goal was to decrease the amount of overfitting indicated by our model metrics above.

4.2 Improved Model Description

From Table 2 and Table 3, it was clear that ResNet18 was the best candidate to build a final model for classification. Although it is computationally the slowest, it is better in terms of accuracy (which is the crucial consideration in our project). With this, we continued experimenting with different hyperparameters to get the best results possible when the official test dataset was provided.

To improve our results with the ResNet18 model, we focused our efforts in two areas: preprocessing of the datasets and adjusting hyperparameters within the model. For preprocessing, we attempted two different strategies to crop each image: in one, we utilized the accompanying JSON file to isolate the section of the image that contained the tumors before resizing each tumor to the same dimensions. With the second approach, we used the same JSON files to separate out the images of the tumor, but they were not resized immediately – instead they were placed in the center of a totally black image. This had the effect of creating an "extended" zero-padded image for each tumor which helped preserve a sense of size and scale between each tumor.

For preprocessing, we also looked at approaches that induced noise inside the training dataset images. We considered a random rotation of each image, a "color jitter" algorithm that randomly adjusted the brightness and color of each picture, and perspective randomizer that would tilt each image at random. We used these preprocessors to "augment" the original dataset, so that our model would train on the original images as well as all of the combined altered images. The intended effect of these preprocessing options was to help reduce the amount of overfitting seen. Other transformations experimented with included grayscaling and inducing normalization on each image. We found that the best combination of these processing options was to enable grayscaling, enable normalization, enable color jitter, enable the perspective randomizer, and leave the random rotation preprocessing option off.

As for hyperparameters, we adjusted batch sizes, weight decay (a regularization technique also aimed at reducing overfitting), learning rates, learning rate scheduling (ensuring that our model did not alter dramatically as the training process matured), type of activation function, and number of epochs trained. Although we found that randomness had a clear effect on our ultimate accuracy metrics, we were able to successfully minimize the amount of overfitting seen in the initial stage of our project.

Hyper Parameter	Value
Resize Dimensions	(250, 250)
Batch Size	64
Weight Decay	0.001
Initial Learning Rate	0.01
Learning Rate Exponential Decay (Gamma)	0.9
Activation Function	reLU
Epochs	60

Table 4: Hyper Parameter Values for the Final Model

The results of our final model (as presented on our test dataset) are below:

	ResNet18
Test Dataset Accuracy	88.8%
Test Dataset Precision	87.5%
Test Dataset Recall	85.2%
Test Dataset F1 Score	86.3%

Table 5: Final results gathered from the final improved ResNet18 model on the test dataset

The accuracy of our model on the train, test, and validation datasets is graphed against the number of training epochs below. At first, the accuracy is unstable; as the epochs progress, the accuracy stabilizes. This is because our learning rate is scheduled to decrease by 10% at each successive epoch. As time goes on, the learning rate decreases and the accuracy stabilizes.

It's also important to note that our preprocessing efforts have effectively closed the gap between the training and validation datasets. This indicates that the overfitting that our original model exhibited has been improved and indicates that the preprocessing settings we have implemented (augmenting the original dataset with transformed images, implementing regularization via weight decay, and padding each cropped image appropriately) is helpful in forcing the model to generalize more effectively. Below, both the training and validation dataset accuracy is nearly 100%.

It should be noted however, that the model's accuracy on the final test dataset is noticeably lower than on the training or validation datasets. Our final model does not generalize well enough to have similar performance on the final test dataset. This indicates that the test data and the original train / validation dataset have some key differences that our final CNN was not able to fully capture. For future work, it would be a good idea to further explore options that will force the CNN to generalize more effectively – this will be discussed in greater detail later on.

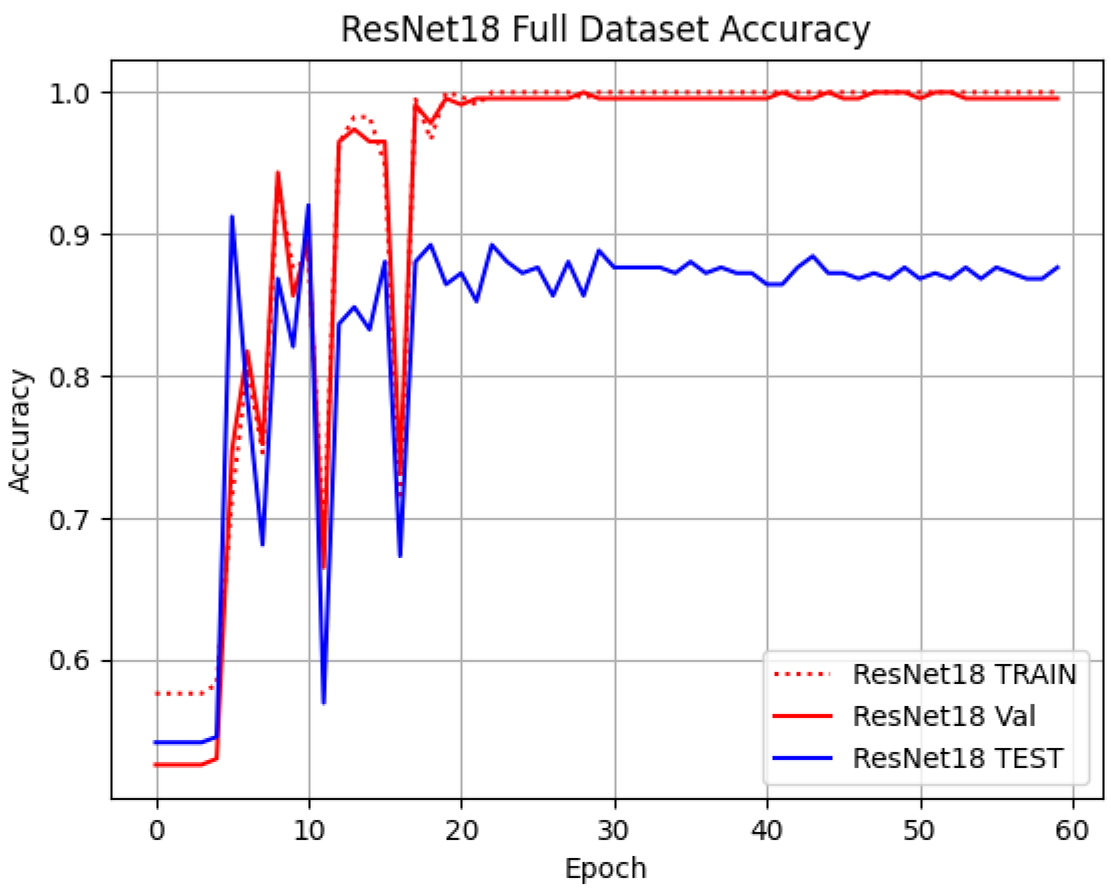


Figure 10: Accuracy record of ResNet18 model vs epoch on training, validation, and test datasets

4.3 Statistical Analysis of Improved Model

4.3.1 AUC, Final Accuracy, and Other Relevant Graphs

To further characterize the effectiveness of our model, we have graphed the ROC curve below. This is a useful visualization that helps show how well the model is able to separate the benign and malignant classes; the model's overall ability to discriminate between the two classes is captured with the AUC (area under curve) metric.

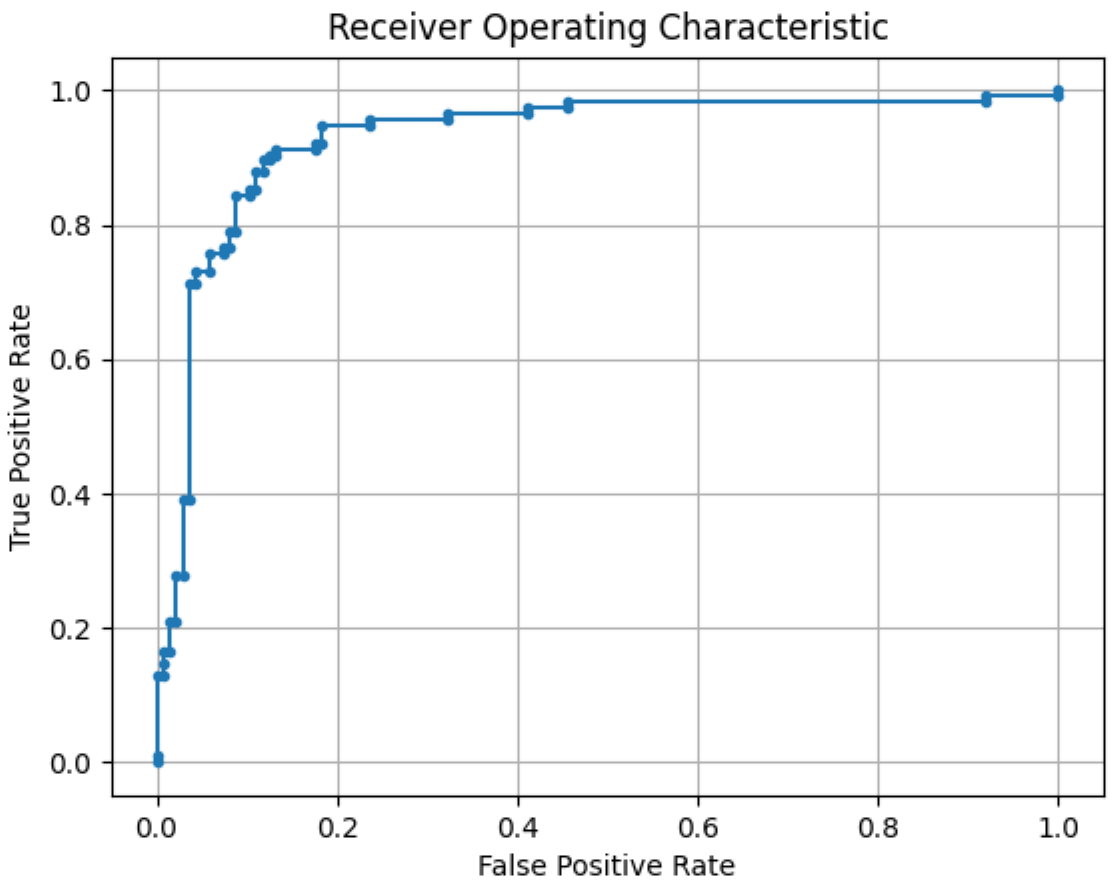


Figure 11: ROC Curve; Calculated AUC is 0.929

The receiver operating characteristic curve indicates that this model is quite good at discriminating between benign and malignant cases – the specific threshold chosen to classify an image as one class or another, however, depends on the application. As the CNN model outputs a probability distribution with the softmax activation function (it assigns a probability to whether or not a particular image is of a benign tumor or a malignant one), it's possible to adjust the probability threshold at which a given image is classified as a benign or a malignant tumor. Not only does this allow us to find a probability threshold that gives us a higher level of accuracy for a given model, but it also allows the user to "tune" the rate of type 1 and type 2 errors. An elaboration on this is offered in the next section with a discussion on the confusion matrix.

4.3.2 Confusion Matrix

Confusion matrices are useful for identifying what types of errors a particular classifier is making (type 1 or type 2 errors, for example). The first confusion matrix shows the final model results assuming a cutoff threshold of 50% (so if the CNN rates a particular image to have 51% probability of belonging to the benign class, it will classify that image as benign). For our confusion matrix, the y-axis shows the true class and the x-axis shows the predicted class according to our model. The labels on each quadrant show the proportion of the actual class that was rated as either benign or malignant by the model (note that each row will sum up to 1). The first confusion matrix shows that the model misclassifies malignant tumors as benign 15% of the time and benign tumors as malignant 10% of the time. When the model classifies a benign tumor as malignant, this is a type 1 error – the opposite is a type 2 error. In this specific application, a type 2 error (the model believes a malignant growth to be benign) is particularly bad because we would not want the model to ignore or miss a potentially malignant growth. It is better to bias this model towards type 1 errors to ensure that even slightly suspicious growths are flagged for further investigation.

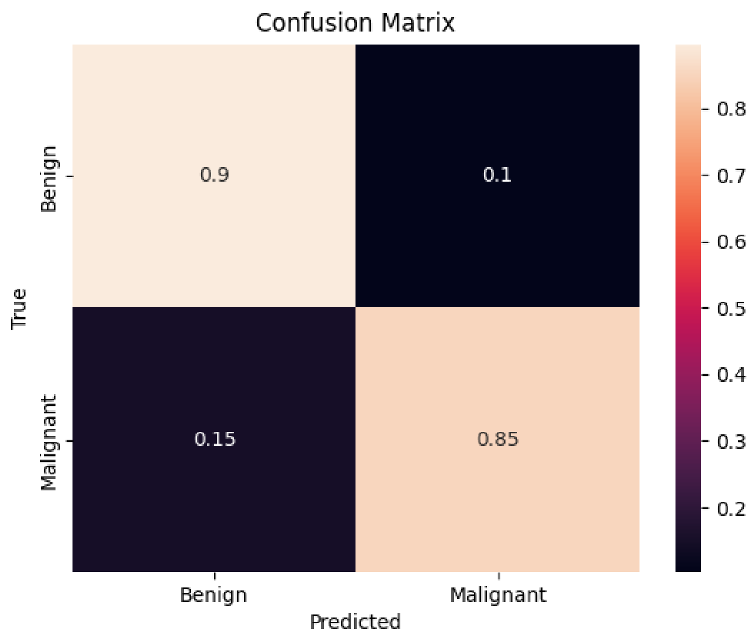


Figure 12: Confusion Matrix; threshold set to 0.5

Because we can adjust the classification threshold as noted above, we can bias the model towards making more type 1 errors instead. The following confusion matrix is with the probability threshold adjusted to maximize overall accuracy – it also shifts the model towards making more type 1 errors than type 2 errors.

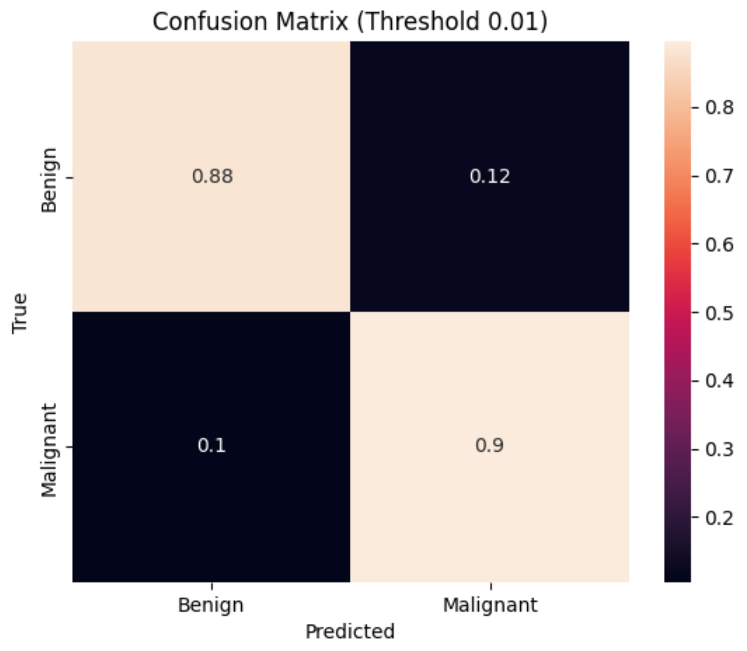


Figure 13: Confusion Matrix; threshold set to 0.01

With this adjusted threshold, our test dataset accuracy improves slightly to 88.8%, and the rate of type 2 errors has decreased.

4.3.4 Baseline Table Statistics

This section discusses the statistics given to us by Dr. Tu. In particular, we are looking to see if there are significant differences between benign and malignant tumor sizes and patient ages.

Characteristics	Benign	Malignant
No. patients	374	212
Age (y)		
Mean ± SD	33.01 ± 12.0	51.9 ± 9.9
Median (range)	32 (58)	50 (61)
Tumor Size/Length (cm)		
Mean ± SD	2.42 ± 1.24	3.14 ± 1.49
Median (range)	2 (8.7)	3 (8.5)
US BI-RADS grade		
2, no.(%)	1.6 %	0.5 %
3, no.(%)	64.4 %	1.4 %
4a, no.(%)	29.1 %	11.8 %
4b, no.(%)	1.9 %	23.6 %
4c, no.(%)	0.3 %	35.9 %
5, no.(%)	0.0 %	23.6 %
No Grade	2.7 %	3.2 %

Table 6: Completed Baseline Table Statistics

To test whether or not the differences in tumor sizes and patient ages between the benign and malignant tumors, we conducted independent two sample t-tests on the mean ages and tumor sizes of both classes. It is important to note that the t-tests we ran were two-sided, so they are only useful in concluding whether or not the means are statistically different. We also ran the t-tests assuming the population variances were equal, which is an assumption that may not apply to reality.

For both tests, the p-value was much lower than an alpha value of 0.01. Therefore, there is sufficient evidence to reject the null hypothesis and conclude that the mean age of benign patients and mean age of malignant patients are significantly different. There is also sufficient evidence to conclude that the mean tumor size of benign patients and the mean tumor size of malignant patients are also significantly different. This suggests that these two features can be used to help discriminate between benign and malignant growths.

The statistical tests are necessary to make these conclusions, as the histograms generated from these statistics can appear inconclusive. For example, a histogram of the tumor sizes by class is shown below:

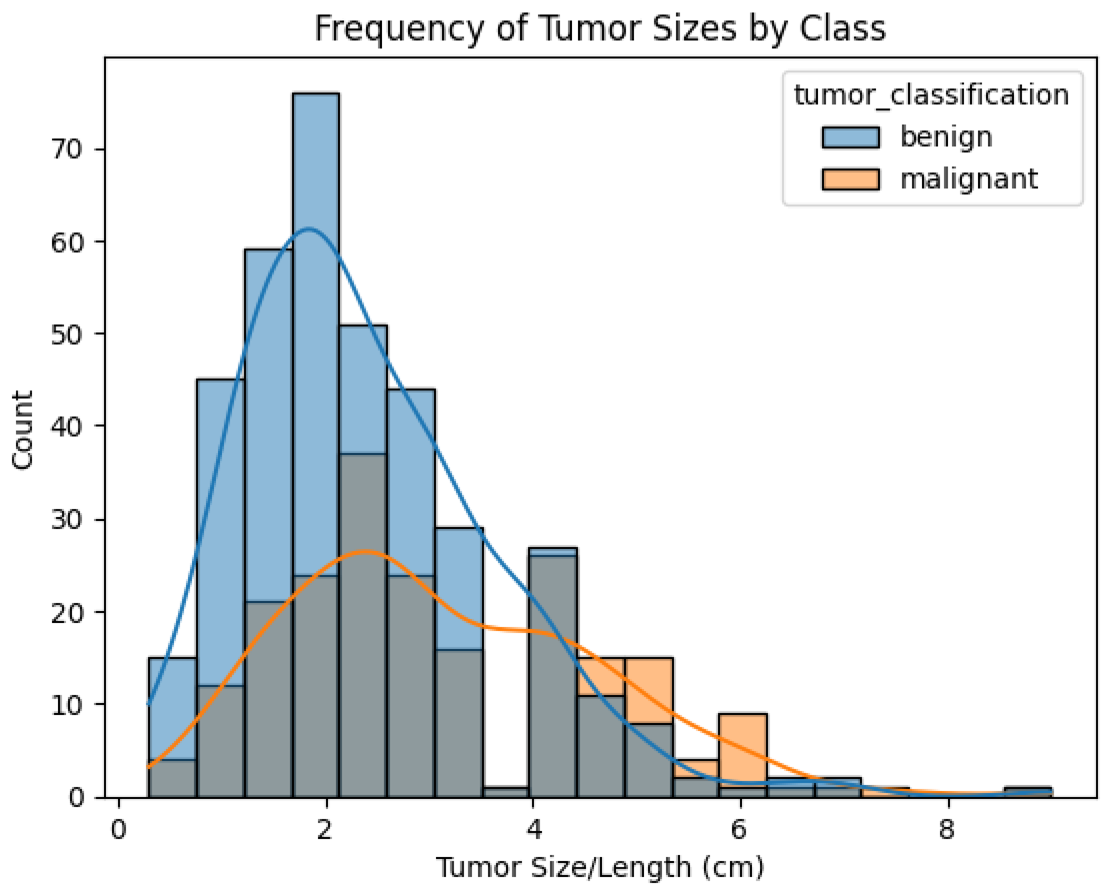


Figure 14: Frequency of Tumor Sizes by Class

Both the benign and malignant tumors appear to follow a similar pattern – they only seem to differ in overall frequency (there are more benign growths in this dataset) and in that the malignant tumor distribution skews very slightly to the right. It is somewhat difficult, however, to discriminate between benign and malignant tumors based on this graph alone. The histogram of patient ages shown below shows a more dramatic difference, however:

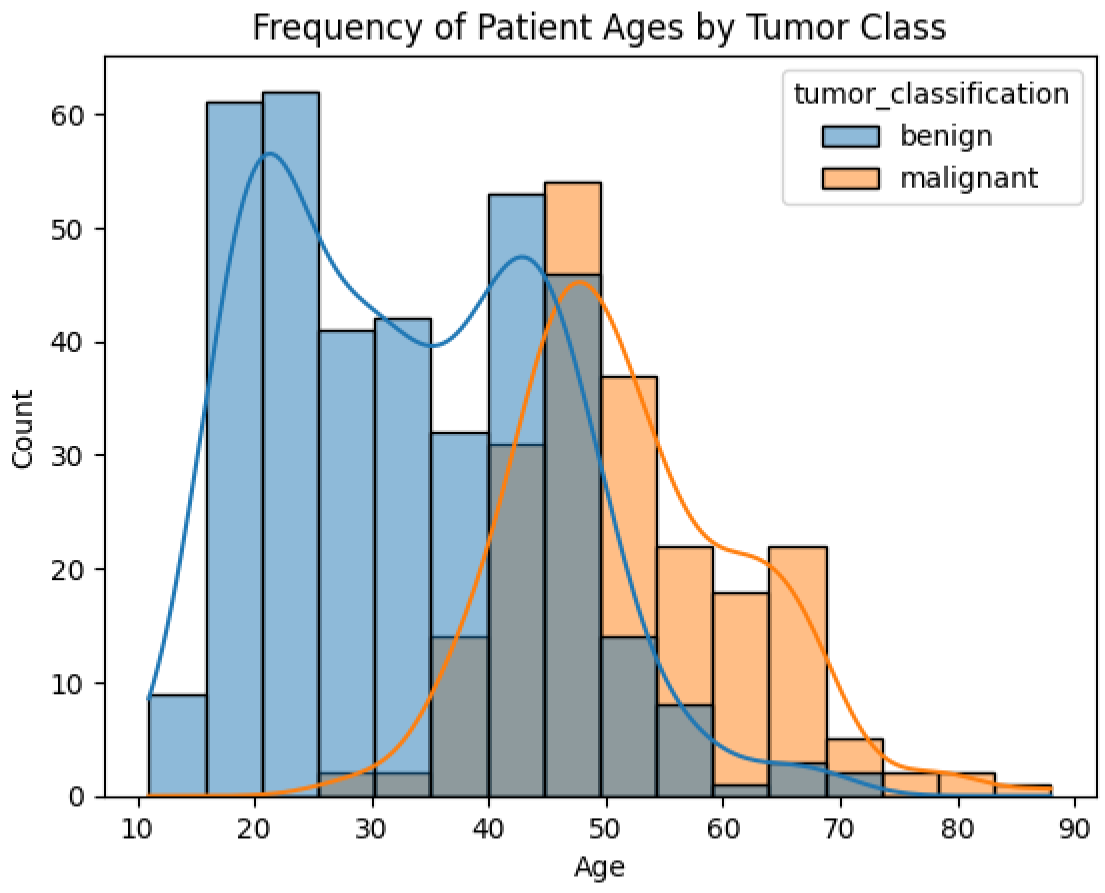


Figure 15: Frequency of Patient Ages by Tumor Class

Here it is very clear that older patients are more susceptible to malignant growths. For both metrics, however, the t-test results should be taken cautiously. As sample sizes increase, even very small differences between different samples can appear to be statistically significant. Therefore we should not rely on statistical analysis alone – looking at the histograms above give the impression that patient age is much more strongly correlated to malignant growths than tumor size alone.

Lastly, we plot a bar graph of the BI-RADS rating as shown in Figure 15. This should give us a good idea on how well our model separates the dataset into benign or malignant.

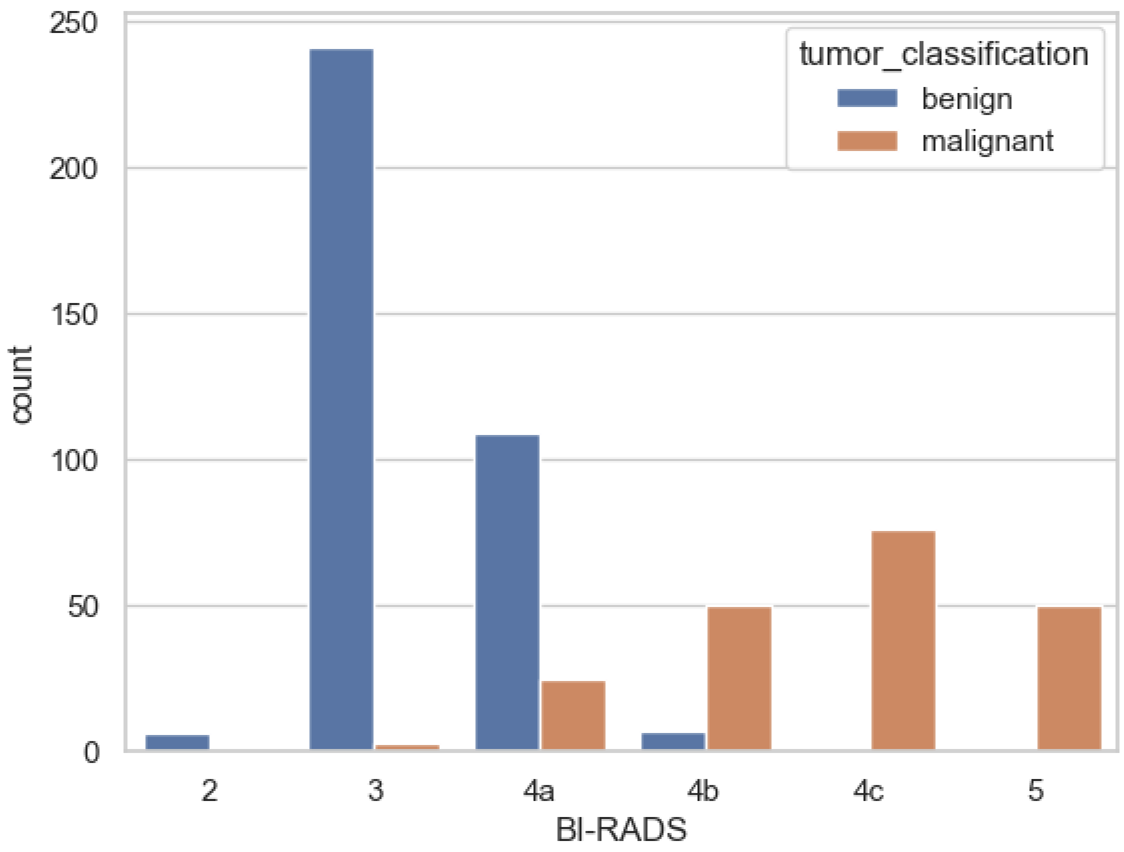


Figure 16: BI-RADS Rating Frequency by Tumor Class

The BI-RADS ratings clearly skew lower for benign tumors and higher for malignant tumors. This means that the ratings do a good job of separating out benign and malignant tumors. Our medical professionals should be applauded!

4.4 Object Detection Discussion

In our Yolov5 model, we used mean average precision (mAP) to assess the performance of object detection models. mAP quantifies the model's ability to precisely identify and locate objects within an image, and it is calculated based on the precision-recall curve obtained from the model's predictions. Precision measures the proportion of correctly predicted objects among all predicted objects (outputted by the Yolov5). Recall measures the proportion of correctly predicted objects among all ground-truth objects (annotated in the dataset). To calculate the precision and recall, the overlap metric, commonly referred to as "intersection over Union" (IoU), is used as the evaluation measure. It is defined as the ratio of the intersection area between the predicted bounding box and the ground truth bounding box to the union area of both boxes. Mathematically, the IoU is calculated as follows:

$$IoU = (Area\ of\ Intersection)/(Area\ of\ Union)$$

Although the IoU threshold is flexible, in our work if the IoU value exceeds 0.5, we consider the predicted bounding box and the ground truth bounding box a match. After obtaining the precision-recall curve, average precision is calculated by computing the area under the precision-recall curve. Finally, the mean average precision is computed by taking the mean of the average precisions from the classes of benign and malignant.

4.4.1 Object Detection Summary for General Tumor Identification on Training Set

First, we utilized YOLOv5 to predict the bounding boxes as given by the manual annotations in the original dataset. Below are 16 randomly selected images with the JSON LabelMe bounding boxes showing the locations of the growths in each image. Since YOLOv5 treats the coordinates of these bounding boxes as a regression problem, we trained the network using the images and the box coordinates directly.

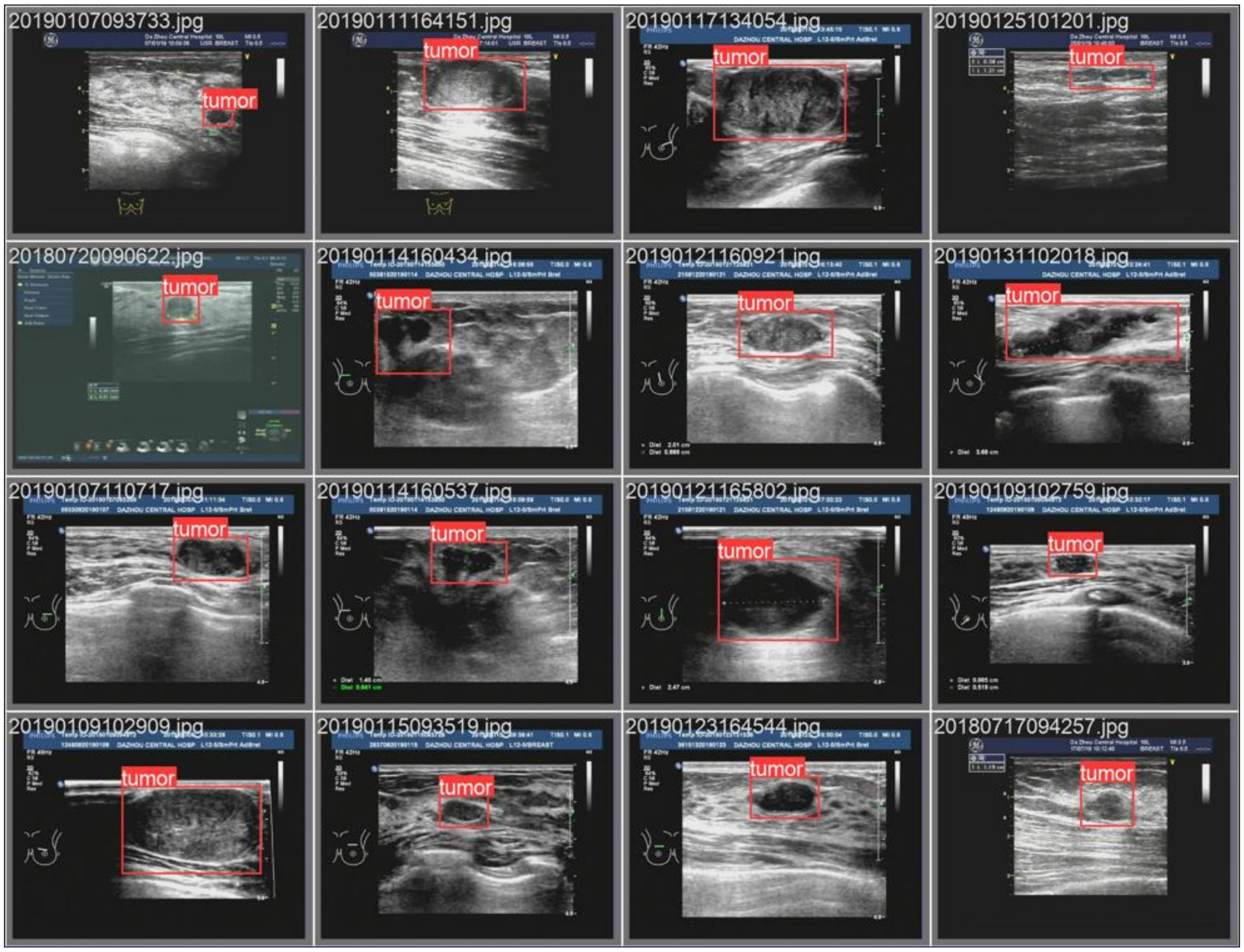


Figure 17: Example of training data with tumors highlighted

After YOLOv5 was trained, we can see the predictions of the tumor locations on the same images as shown above. The red bounding boxes show the predictions, and the number above the box indicates YOLOv5's confidence that the bounding box contains a tumor. From a cursory glance, one can see that the YOLOv5 algorithm did quite well with the identification. Farther below we have calculated metrics quantifying the algorithm's performance on the training dataset.

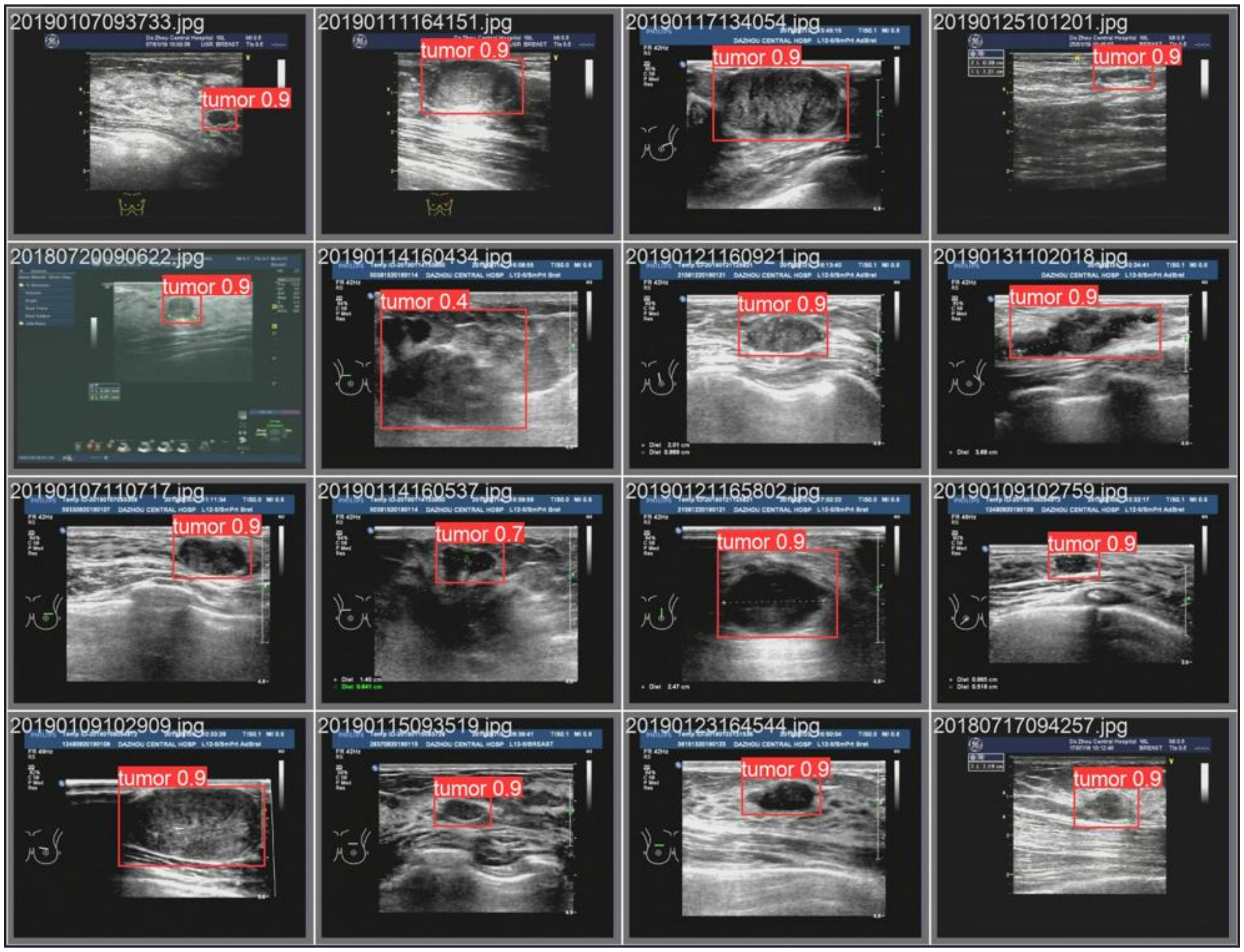


Figure 18: Example of training data with YOLOv5 predictions (numbers are confidence scores)

For the results below, the original image dataset was split into a purely train dataset (80% of the original dataset) and a validation dataset (20% of the dataset). The model was then trained – the table below shows the ultimate metrics on interest on the validation dataset.

Metric for Object Detection	YOLOv5 Score on Validation Dataset
Precision	97.4%
Recall	96.6%
mAP 0.5	98.7%

Table 7: Final results gathered from the original dataset with YOLOv5

For the results below, the original image dataset was split into a purely train dataset (80% of the original dataset) and a validation dataset (20% of the dataset). The model was then trained – the table below shows the ultimate metrics on interest on the validation dataset.

The metrics above indicate that the YOLOv5 algorithm performs admirably. Precision and recall are defined as per usual, with precision measuring the proportion of true positives among all positive predictions and recall measuring the proportion of true positives among all actual positive cases. In this context, precision and recall are scoring whether or not the predicted bounding box sufficiently overlaps the original hand-annotated bounding box. The mAP (mean average precision) at an IoU (intersection over union) of 0.5 is also quite high – recall that it summarizes how well the predicted bounding box overlaps the original ground-truth bounding box. Judging by these results and by the sample of images above we can conclude that the YOLOv5 algorithm performs very well.

Below are additional figures that help quantify the performance of the YOLOv5 algorithm in identifying masses in ultrasound images. The precision-recall curve shown below is similar to the ROC curve above in that it helps quantify how well a particular classifier operates on a binary classification task.

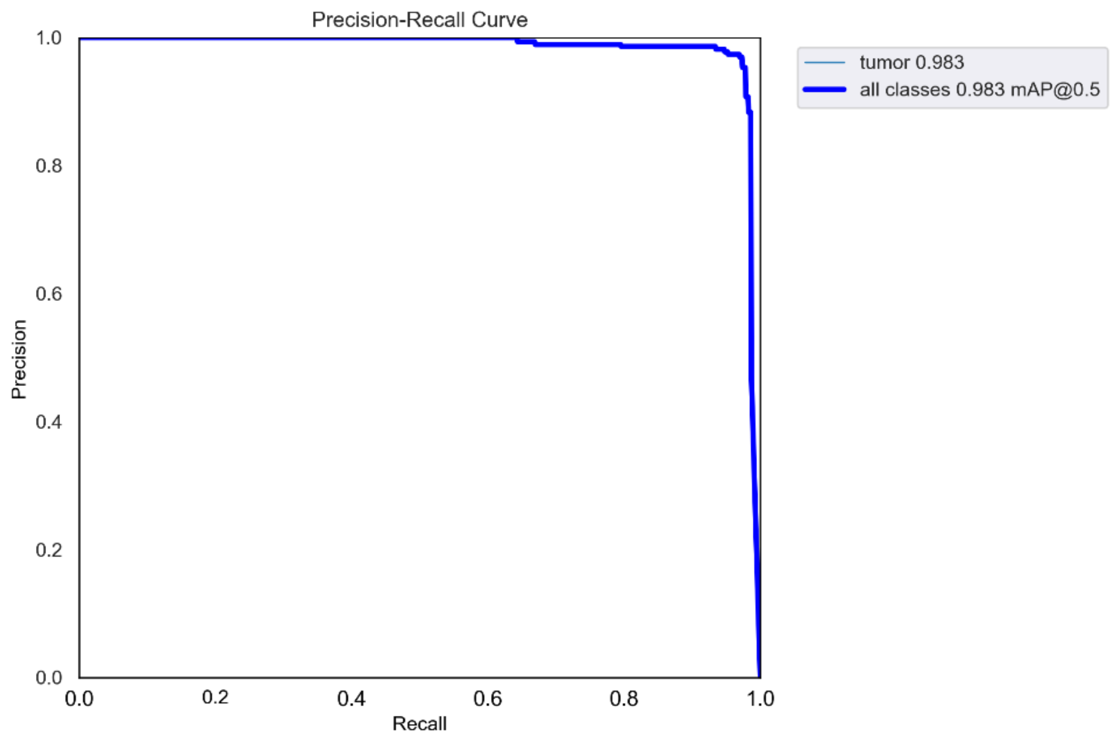


Figure 19: Precision Recall Curve for YOLO on Training Dataset

Above, we can see that the classifier is nearly perfect as there is very little tradeoff between precision and recall in our classifier.

Below is an F1-Confidence curve. The F1 score can be thought of a combination of recall and precision, so it's an ideal summarization metric. Confidence is a YOLOv5 specific metric – as explained by the original paper (Redmon, Divvala, Girshick, & Farhadi, 2016):

"Our system divides the input image into a S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.

Formally we define confidence as Pr(Object) ∗ IOUtruth pred . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth."

Here is the F1-Confidence curve:

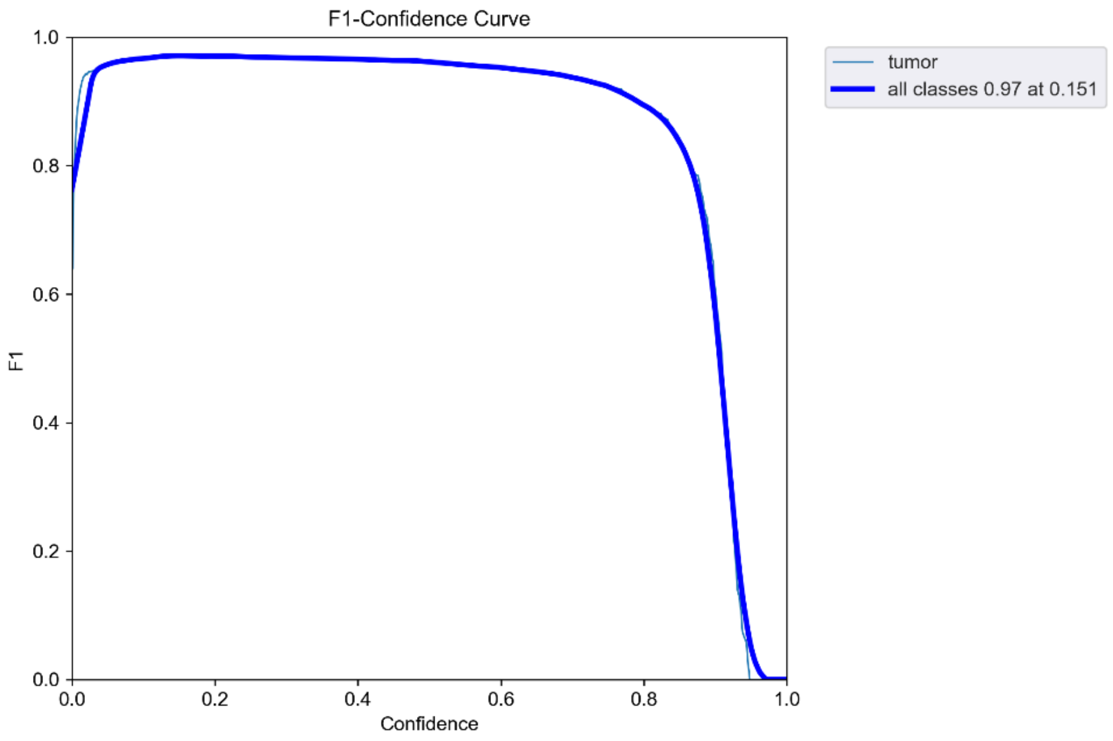


Figure 20: F1 Score vs Confidence Curve for YOLO on Training Dataset

When the confidence threshold is low, precision is also low (leading to a low F1 score). When the confidence threshold is very high, recall suffers and the F1 score also decreases. This graph shows how the model's confidence forces a tradeoff between both precision and recall.

The last graph below is a training record of the model graphed against each epoch. The graphs below show some opportunity to increase the number of epochs, but they also indicate that additional training may result in some overfitting (notices that the validation dataset's obj_loss metric is slowly increasing as epochs increase).

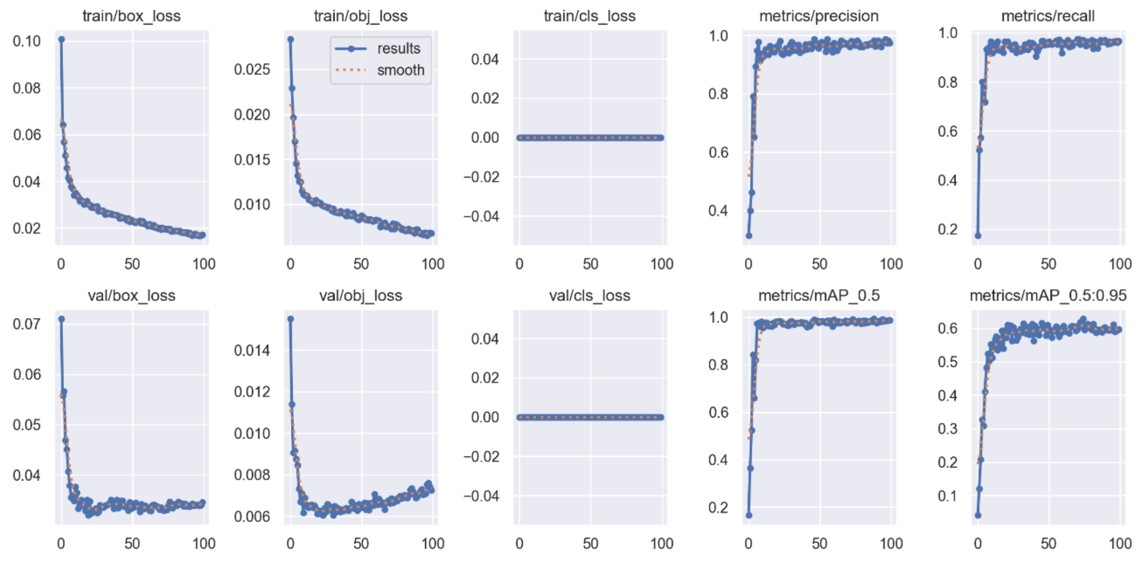


Figure 21: Training Results vs Epoch

4.4.2 Object Detection Summary for General Tumor Identification on Test Set

Now that the model performance has been successfully characterized on the training and validation datasets, it can now be evaluated using the final test dataset given to us by AI+ First. The format of this section is identical to that of the preceding section, but the metrics are now focused on the YOLOv5 model's performance on the final test dataset.

Metric for Object Detection	YOLOv5 Score on Test Dataset
Precision	85.0%
Recall	82.7%
mAP 0.5	81.6%

Table 8: Final results gathered on the test dataset with the YOLOv5 model

We can immediately note that the metrics are slightly degraded. Like our results with the ResNet18 model, this shows that the model is not doing an optimal job of generalizing its predictions from the original training datasets.

For comparison's sake, the F1-confidence curve and the precision-recall curve describing the model's performance on the test dataset follow. We can also see signs of degraded performance here as well.

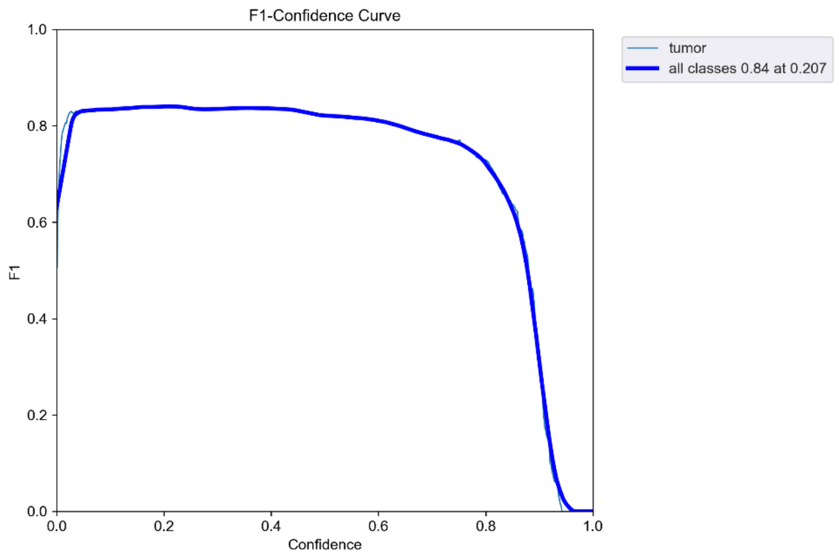


Figure 22: F1-Confidence curve for general tumor detection

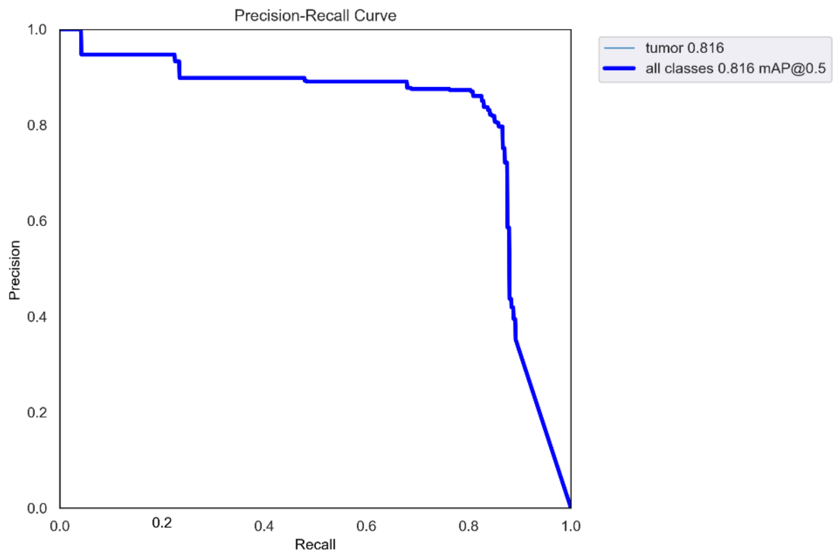


Figure 23: Precision-Recall curve for general tumor detection

4.4.3 Object Detection Summary with Benign / Malignant Classification on Training Set

When working with YOLOv5, we noticed that the CNN does not only output predictions on bounding boxes for object detection. It can also classify local features in images as well, and with this in mind we attempted to train the model to also classify each growth.

Below are the main metrics (the same as in the preceding sections) that summarize the model's performance, but averaged over both the benign and malignant classes. We can see that the metrics are still quite good, but not as high as indicated in the pure object detection task. This indicates that it is more difficult to make the YOLOv5 model classify and detect at the same time.

Metric for Object Detection	YOLOv5 Overall Score [Benign and Malignant Classes]
Precision	94.0%
Recall	95.1%
mAP 0.5	97.0%

Table 9: Final results gathered on the training dataset with the YOLOv5 model with both classes

The same collection of images shown above are redisplayed, but with the proper tumor classification in place. We can see that the YOLOv5 model still goes a good job at correctly identifying most of the growths, but there are noticeable deficiencies.

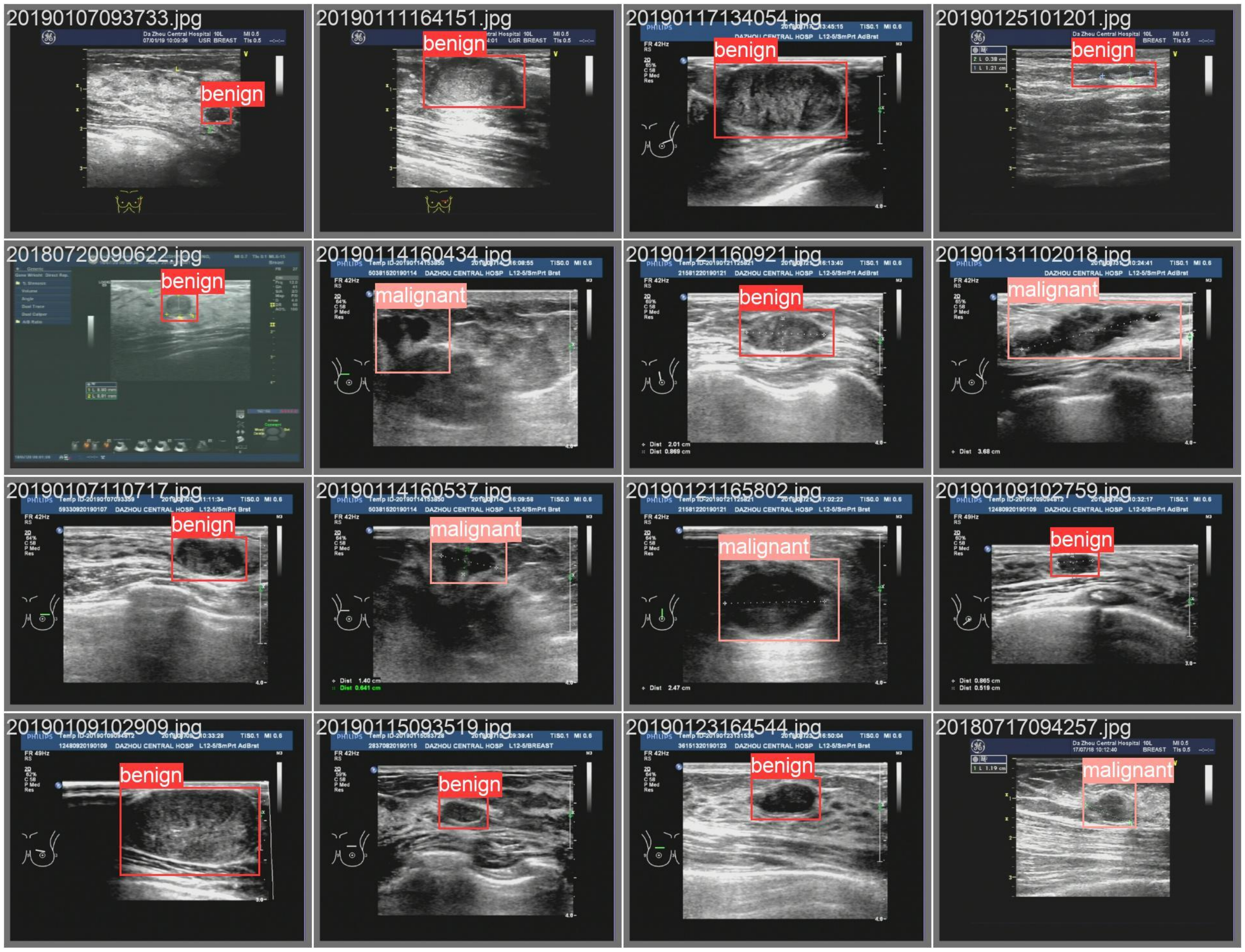


Figure 24: Example of training data with classes indicated

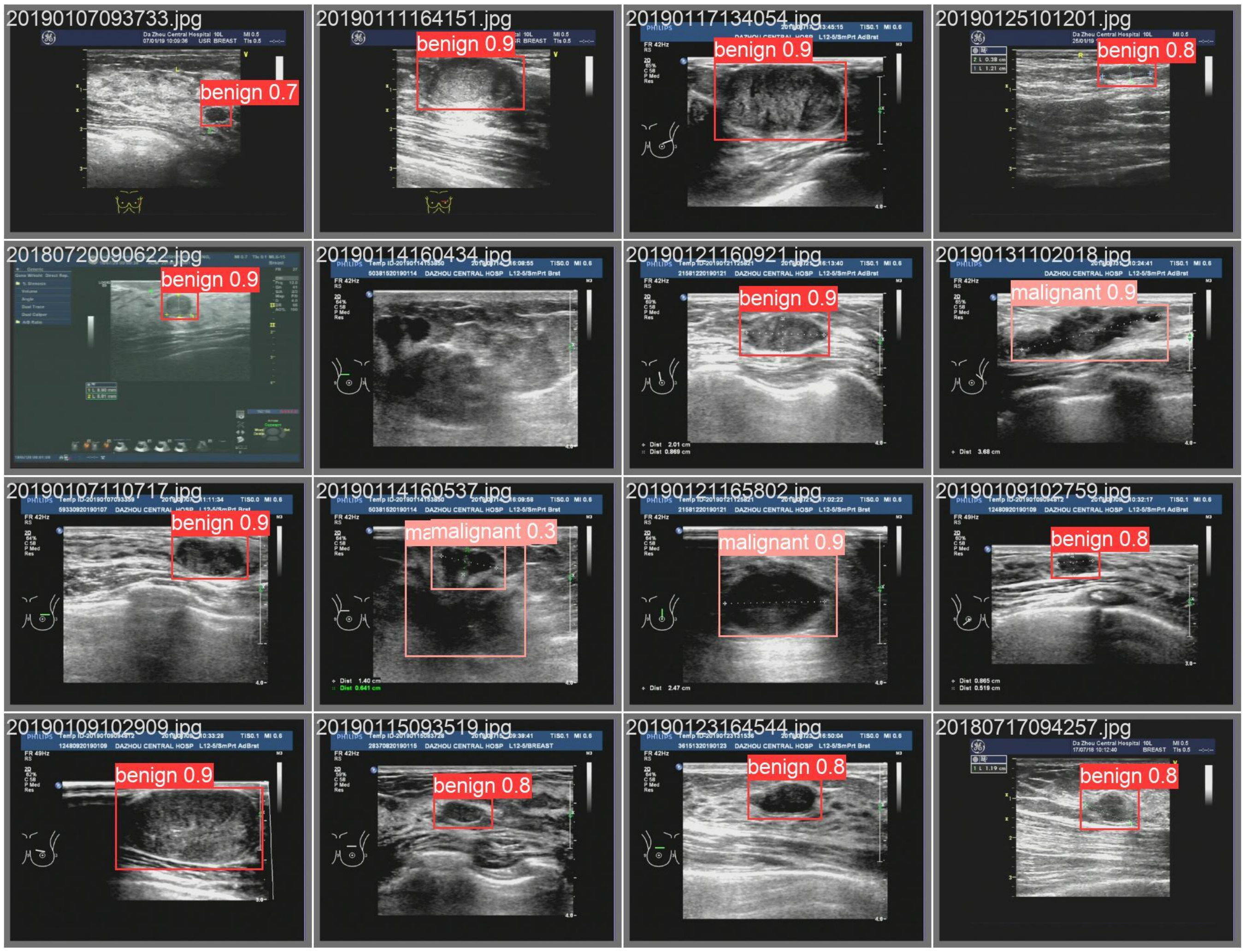


Figure 25: Example of training data with YOLOv5 class predictions

The precision-recall and F1-confidence curves are also shown here. They indicate strong performance but demonstrate some loss when compared against the general object detection class where the model is not asked to differentiate between benign and malignant tumors. From the higher precision-recall and F1-confidence curves, we can state that the model has an easier time correctly identifying benign tumors.

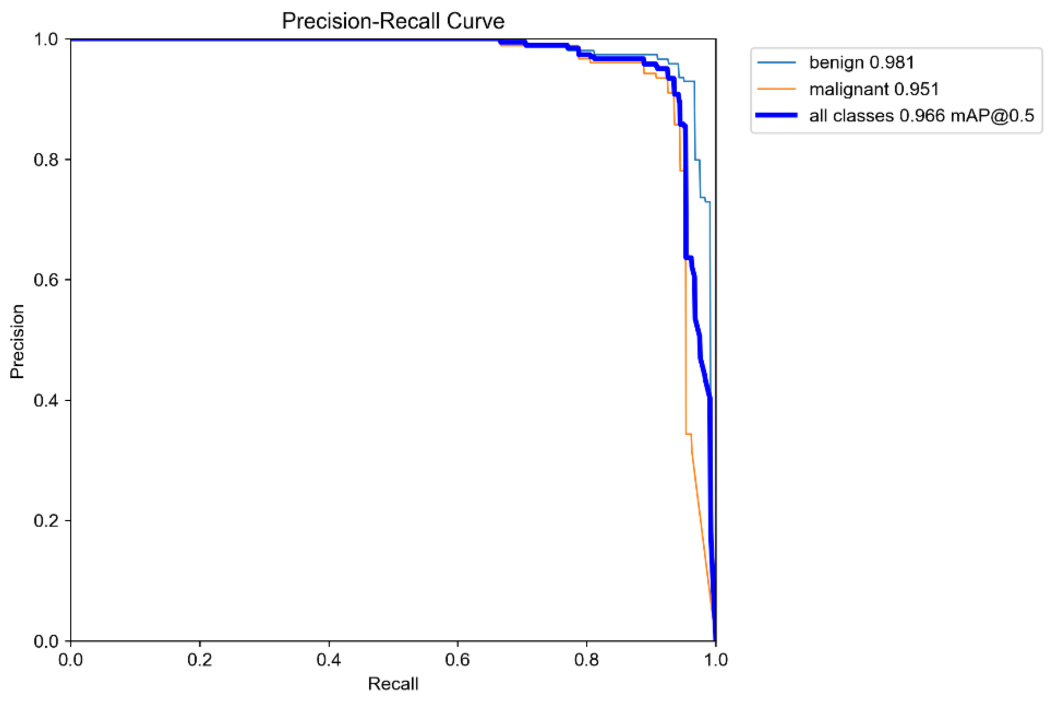


Figure 26: Precision-Recall curve for YOLOv5 Classification

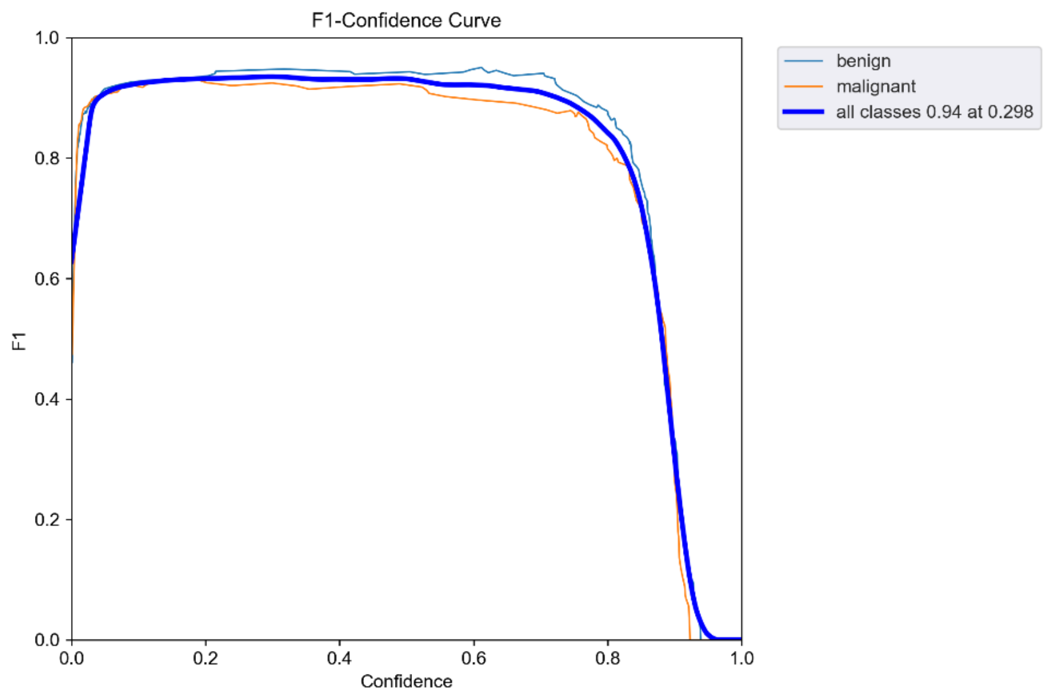


Figure 27: F1-Confidence curve for YOLOv5 Classification

The training log for the classification and object detection task is shown below. The observed patterns are very similar to the first log with only one class, indicating that we can likely train for more epochs while still being cautious of overfitting.

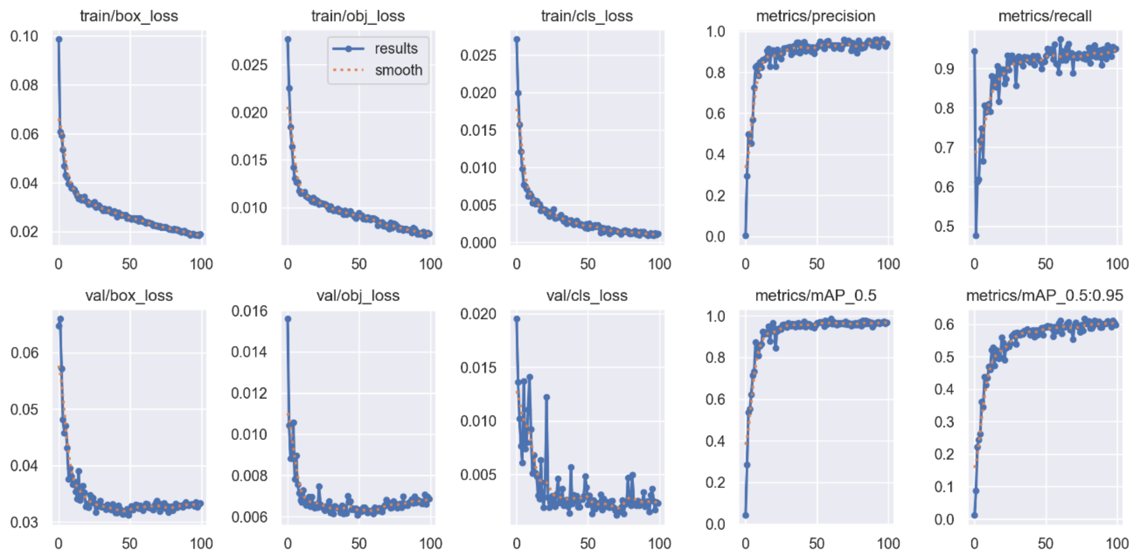


Figure 28: YOLOv5 training history for classification task

4.4.4 Object Detection Summary with Benign / Malignant Classification on Test Set

Now we characterize the effectiveness of the YOLOv5 model on the test dataset. We have separated the metrics for each class in the table below.

Metric	Class	YOLOv5 Overall Scores [Benign / Malignant Classes]
Precision	Benign	77.5%
Recall		75.2%
mAP 0.5		71.0%
Precision	Malignant	92.4%
Recall		90.7%
mAP 0.5		95.7%
Precision	Overall	85.0%
Recall		82.9%
mAP 0.5		83.3%

Table 10: Final results gathered on the test dataset with the YOLOv5 model with both classes

Notice we perform worse on benign tumors! Note that the classification metrics are good with the malignant class, but comparatively poor on the benign class. This indicates that our model specifically struggles with classifying benign tumors. This observation is further supported by the figures below.

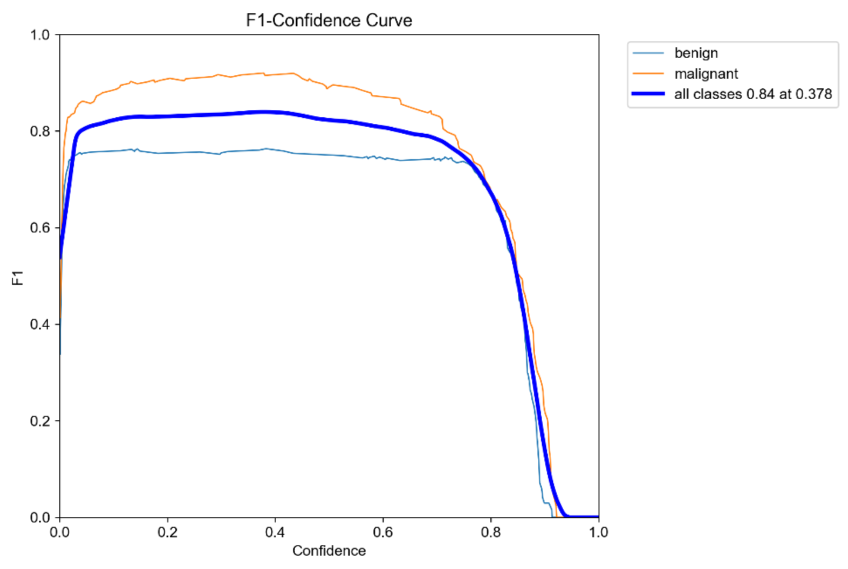


Figure 29: F1-Confidence curve for YOLOv5 classification on test dataset

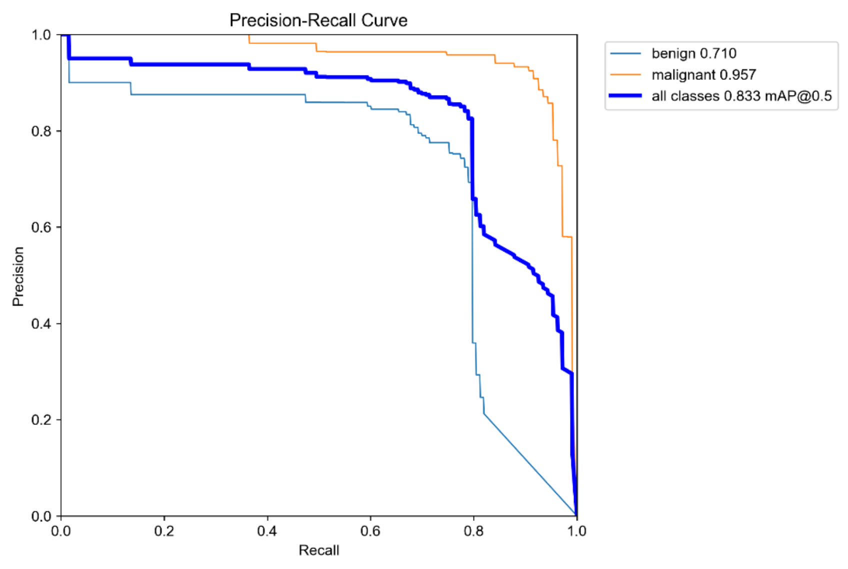


Figure 30: Precision-Recall curve for YOLOv5 classification on test dataset

In both the precision-recall curve and F1-confidence curve above, note that our classifier specifically struggles with the benign class in general. This is something that will require more investigation into the benign tumors inside the test dataset, but we do know that our models have trouble generalizing our training dataset well enough to perform as effectively on the testing dataset.

Finally, we can use YOLOv5 to identify the tumors in each image in the classification sense – so far, our precision and recall metrics have been in the language of object detection and overlapping bounding boxes. We can now go back to discussing the classic definitions of accuracy, precision, recall, and F1 score to help compare

YOLOv5's classification effectiveness against the ResNet18 results. Below, we simply observe whether or not the YOLOv5 model has identified the correct tumor classification for each image in the test dataset.

	YOLOv5 Scores on the Test Dataset
Test Dataset Accuracy	90.0%
Test Dataset Precision	96.1%
Test Dataset Recall	95.1%
Test Dataset F1 Score	95.6%

Table 11: Final results gathered from the final YOLO model on the test dataset

The accompanying confusion matrix is shown below. The overall accuracy is quite high, and the model is well-balanced as well with no propensity towards type 1 or type 2 errors (this last point may be tuned to user specification, however).

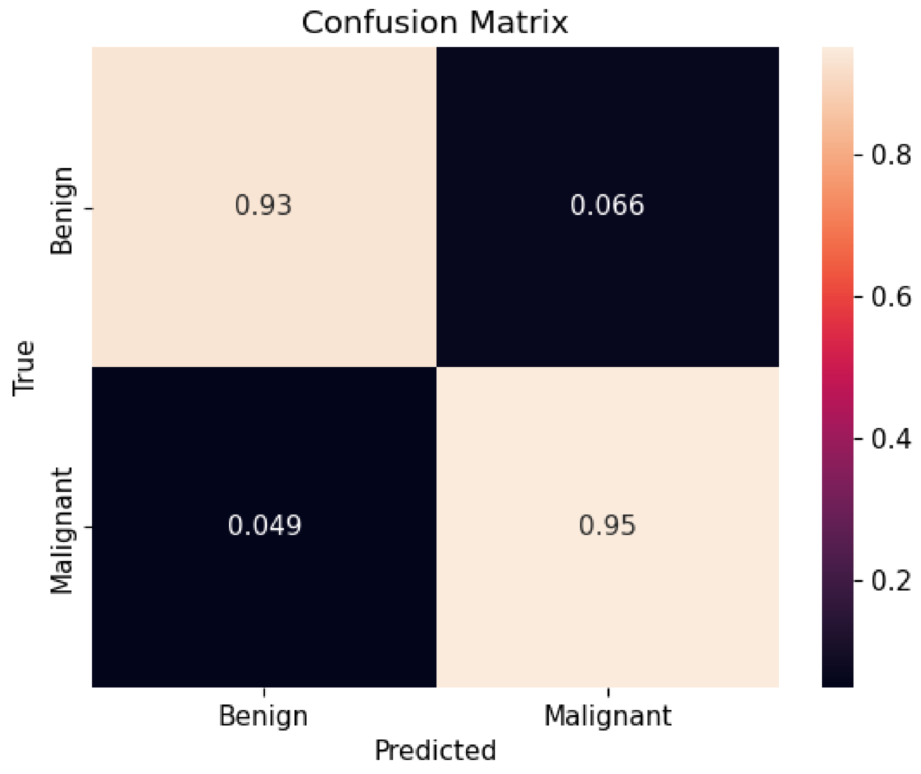


Figure 30: YOLOv5 classification confusion matrix

The YOLOv5 model presented here outperforms the other base CNN architectures we tested throughout the rest of the project when it comes to pure prediction accuracy on the test dataset. Given that this model can perform both the object detection and classification with a high degree of accuracy, we recommend that this model type be used.

5 Future Work

Due to time constraints, our team was able to experiment and produce a final model with only the ResNet18 model. Should our team have more time, exploration of other hyperparameters for DenseNet 121, MobileNet v1, and MobileNet v2 may produce fruitful results. It should be noted however, that running these models requires a strong machine with a good GPU to run quickly (even with MobileNet). Running an exhaustive DoE with many different hyperparameters would be extremely time consuming. Additionally, some of the baseline models our team built already had reasonable (but not optimal) results when tested on the provided training dataset, and a few extra tweaks to the model may have also performed as well as the final ResNet18 model. Furthermore, having a separate structure for building the MobileNet v2 code (without short time constraints) may avoid the reshaping issue we came across. As mentioned in theory, MobileNet v2 should outperform MobileNet v1 in terms of accuracy as well as computation time but it was not shown in our project.

Extra time would also help tremendously for our team to get higher accuracy for our final ResNet18 model as well. Additional preprocessing options, hyperparameter optimization, and even the usage of a genetic algorithm to help discover and refine a better CNN architecture were all considerations for future work on this project.

The YOLOv5 model turned out to be the best, and we recommend exploring this model type more deeply for this specific task. It has good classification accuracy, and its ability to perform both classification and object detection at the same time makes it easier to use. Another benefit is that YOLOv5 can work very quickly, allowing for "live" applications where the classifier is working with real-time images. This may become a more attractive feature if live-service applications are desired.

6 Conclusion

In this report, our team introduced the dataset and its background as well as explored the models we built to classify the dataset. Out of the 3 models (ResNet18, DenseNet121, and MobileNet v1), ResNet18 performed the best against the provided training dataset as well as a randomly split validation set from the training dataset. Additionally, our team decided to experiment with MobileNet v2 as well and compare it with MobileNet v1. We discovered that the unified structure of our code may have caused MobileNet v2 to not work as expected and decided not to pursue this approach further. As a result, ResNet18 was chosen as our final model due to time constraints and experiment with some hyperparameters to achieve a reasonable outcome.

We found that the final improved ResNet18 model produced a maximum accuracy of around 88%. We also found that the high accuracies regarding ResNet18 were due to overfitting and our team worked to fix this issue while experimenting with hyperparameters. The best results came from augmenting the dataset with altered images which allowed our team to fix the overfitting issue and work better with the sponsor-provided test dataset.

The test dataset was also found to have some randomness that differs from those in the initial training dataset. Additionally, our team theorizes that the test dataset may have come from a different machine or time stamp than those in the initial training dataset. These differences may indicate that more data may be required in order for better classification than our team already has. Furthermore, the final ResNet18 performed well in classifying the tumors based on benign and malignant but further testing of thresholds may be needed to avoid Type I or Type II errors in the future.

That said, the YOLOv5 model gave the best overall classification results and performed impressively on its object detection tasks. As it is also open-source, we recommend that this model type be considered for future work.

Overall, our project has reasonable results as of today but still has some ongoing experimental designs. Some future considerations such as experimenting with more hyperparameters and working on improving the other models (DenseNet121, MobileNet v1, MobileNet v2) may help in this classification discussion as well. The YOLO models also have several variants to explore as well, though they may be more resource intensive.

We are grateful to our sponsors at AI+ First for allowing us to work on such a rewarding project – we were very impressed with the quality of the data gathered, and hope the insights assembled here can help with the ongoing work. Special thanks go to Dr. Tu for helping all of the students with her prompt and informative replies. We are looking forward to your continued excellence and hope for your future success!

References

Brattain, L. J., Telfer, B. A., Dhyani, M., Grajo, J. R., & Samir, A. E. (2018, February). Machine learning for medical ultrasound: status, methods, and future opportunities. *Abdominal Radiology*, 786-799. doi:https://doi.org/10.1007/s00261-018-1517-0

Cancer.Net Editorial Board. (2023, 02). *Breast Cancer: Statistics*. Retrieved from Cancer.Net: https://www.cancer.net/cancer-types/breast-cancer/statistics

Devolli-Disha, E., Manxhuka-Kërliu, S., Ymeri, H., & Kutllovci, A. (2009). Comparative accuracy of mammography and ultrasound in women with breast symptoms according to age and breast density. *Bosnian Journal of Basic Medical Sciences*, 131-136. Retrieved from https://doi.org/10.17305/bjbms.2009.2832

Giger, M. L., Al-Hallaq, H., Huo, Z., Wolverton, D. E., & Chan, C. W. (1999, November). Computerized analysis of lesions in US images of the breast. *Academic Radiology, 6*(11), 665-674. Retrieved from https://doi.org/10.1016/S1076-6332(99)80115-9

Han, S., Kang, H. K., Jeong, J. Y., Park, M. H., Kim, W., Bang, W. C., & Seong, Y. K. (2017). A deep learning framework for supporting the classification of breast lesions in ultrasound images. *Physics in medicine and biology, 62*(19), 7714-7728. Retrieved from https://doi.org/10.1088/1361-6560/aa82ec

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 770-778). Las Vegas. Retrieved from https://ieeexplore.ieee.org/document/7780459

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (n.d.). MobileNets: Efficient Convolution Neural Networks for Mobile Vision Application. *arXiv preprint*. Retrieved from https://arxiv.org/abs/1704.04861

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 2261-2269). Honolulu.

Kalafi, E., Jodeiri, A., Setarehdan, S., Lin, N., Rahmat, K., Taib, N., . . . Dhillon, S. (2021). Classification of Breast Cancer Lesions in Ultrasound Images by Using Attention Layer and Loss Ensemble in Deep Convolutional Neural Networks. *Diagnostics, 11*(10), 1859. Retrieved from https://pubmed.ncbi.nlm.nih.gov/34679557/

Kramer, J. (2012, 5 2). *Datum Corp.* Retrieved from cancer.net: www.hey.com

Rouhi, R., Jafari, M., Kasaei, S., & Keshavarzian, P. (2015). Benign and Malignant Breast Tumors Classification Based on Region Growing and CNN Segmentation. *Expert Systems with Applications, 42*(3), 990-1002. Retrieved from https://www.sciencedirect.com/science/article/abs/pii/S0957417414005594

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. a*rXiv preprint.* Retrieved from https://arxiv.org/abs/1801.04381v4

Simonyan, K., & Zizzerman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint*. Retrieved from https://arxiv.org/abs/1409.1556

Wei, M., Du, Y., Wu, X., Su, Q., Zhu, J., Zheng, L., . . . and Zhuang, J. (2020). A Benign and Malignant Breast Tumor Classification Method via Efficiently Combining Texture and Morphological Features in Ultrasound Images. *Computational and Mathematical Methods in Medicine*. Retrieved from https://pubmed.ncbi.nlm.nih.gov/33062038/

Yi, D., L., S. R., Ill, D. C., Dunnmon, J., Lam, C., Xiao, X., & Rubin, D. (2017). Optimizing and Visualizing Deep Learning for Benign/Malignant Classification in Breast Tumors. *arXiv preprint*. Retrieved from https://arxiv.org/abs/1705.06362

Zhang, J., Wu, C., Yu, X. Y., & Lei, X. L. (2021). A Novel DenseNet Generative Adversarial Network for Heterogenous Low-Light Image Enhancement. *Front. Neurorob., 15*. Retrieved from https://www.frontiersin.org/articles/10.3389/fnbot.2021.700011/full

Zhang, Q., Xiao, Y., Dai, W., Suo, J., Wang, C., Shi, J., & Zheng, H. (2016). Deep learning based classification of breast tumors with shear-wave elastography. *Ultrasonics, 72*, 150-157. Retrieved from https://pubmed.ncbi.nlm.nih.gov/27529139/