Contents

Final Report

MDS 2021 Capstone Project with the Gerontology Diabetes Research Lab (GDRL)

By: Ela Bandari, Lara Habashy, Javairia Raza and Peter Yang

1. Executive Summary

Subclinical lipohypertrophy is traditionally evaluated by visual inspection or palpation. Recent work has shown that lipohypertrophy may be detected by ultrasound imaging [Kapeluto *et al.*, 2018]. However, the criteria used to classify lipohypertrophy using ultrasound imaging is only familiar to and implemented by a small group of physicians [Madden, 2021]. In an effort to improve the accessibility and efficiency of this method of detection, we have developed a supervised machine learning model to detect lipohypertrophy in ultrasound images.

We developed an optimal machine learning model through an iterative process of data augmentation, hyperparameter tuning, and architecture selection. The final optimal model accurately classified 76% of unseen test data. We tested different image augmentation techniques and ultimately decided on adding random flipping, contrast and brightness. We then fit a variety of pre-existing model architectures trained on thousands of images to our augmented dataset. We optimized the parameters by which the model learned and then carefully examined the different models' performance and fine tuned them to our dataset before selecting the best performing model. We have made our best performing model accessible to potential users via a web interface. Our work has demonstrated the potential that supervised machine learning holds in accurately detecting lipohypertrophy; however, the scarcity of the ultrasound images has been a contributor to the model's shortcomings. We have outlined a set of recommendations (see section 6. Conclusions and recommendations) for improving this project; the most notable of which is re-fitting the model using a larger dataset.

2. Introduction

Subclinical lipohypertrophy is a common complication for diabetic patients who inject insulin. It is defined as the growth of fat cells and fibrous tissue in the deepest layer of the skin following repeated insulin injections in the same area. It is critical that insulin is not injected into areas of lipohypertrophy as it reduces the effectiveness of the insulin [Kapeluto *et al.*, 2018]. However, recent research by Kapeluto *et al.* [2018] has found ultrasound imaging techniques are more accurate in finding these masses than a physical examination of the body by a healthcare professional. Examples of these images are shown in Fig. 1 below. Unfortunately, the criteria to classify lipohypertrophy using ultrasound imaging is only implemented by a small group of physicians. To expand the usability of this criteria to a larger set of healthcare professionals, the capstone partner is interested in seeing if we can leverage supervised machine learning techniques to accurately classify the presence of lipohypertrophy given an ultrasound image.

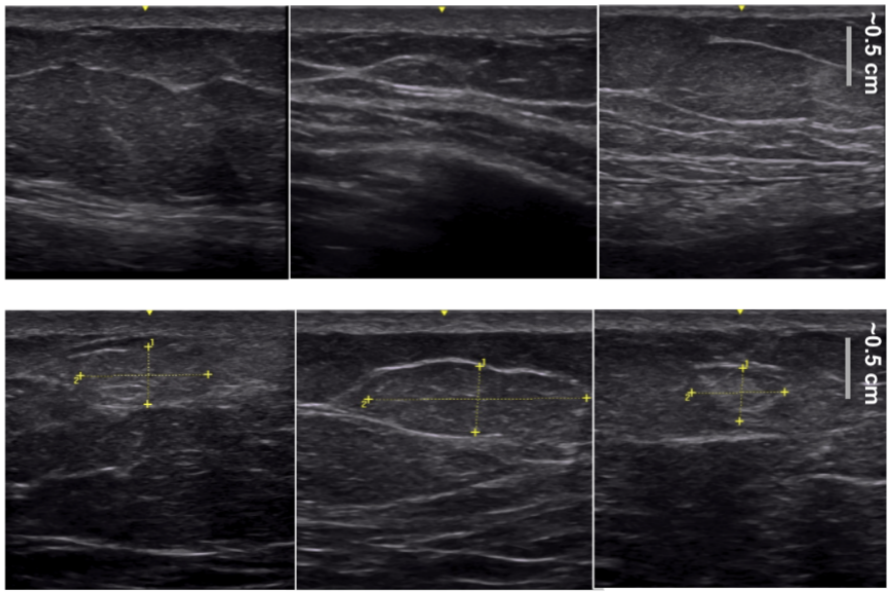

Fig. 1 Some examples of images found in our dataset. Top row is negative (no lipohypertrophy present) and bottom row is positive (lipohypertrophy present) images where the yellow annotations indicate the exact area of the mass.

Our current dataset includes 218 negative images (no lipohypertrophy present) and 135 positive images (lipohypertrophy present) as shown in Fig. 2 below. Notably, this is considered to be a very small dataset for a deep learning model.

Our specific data science objectives for this project include:

1. Use the provided dataset to develop and evaluate the efficacy of an image classification model. Some of our specific goals include:
    1. Data Augmentation: applying random image transformations as to expland the dataset (section 4.1 Data Preparation)
    2. Transfer Learning: using a model that has been pre-trained on thousands of images and using it for our dataset (section 4.2 Model Development)
    3. Optimization of Learning: figuring out what are the best parameters that will make the model most optimal for learning the data (section 4.2 Model Development)
    4. Object Detection: training a model to detect the location of lipohypertrophy on an image (section 4.3 Object Detection)
2. Deploy the model for a non-technical audience (section 5. Data Product and Results).

	Positive	Negative
Count	135	218
Proportion	38.0%	62.0%

Fig. 2 The total number of positive and negative examples in our dataset.

3. Literature Review

Throughout the project, the team has consulted literature to gain insight and direction on current practices relevant to our problem and our dataset to guide and validate our decision-making. Below is a summary of the relevant findings.

3.1 Similar Work

Prediction of masses in ultrasound images using machine learning techniques has been an ongoing effort in clinical practice for the past few decades. To assist physicians in diagnosing disease, many scholars have implemented techniques such as regression, decision trees, Naive Bayesian classifiers, and neural networks on patients' ultrasound imaging data [Huang *et al.*, 2018]. Further, many studies involving ultrasound images have preprocessed the images

to extract features. This study by Chiao *et al.* [2019] shows that CNNs using ultrasound images perform better than radiomic models at predicting breast cancer tumours. Another recent study shows success in classifying liver masses into 1 out 5 categories with 84% accuracy using a CNN model [Yasaka *et al.*, 2018].

Furthermore, recent research has delved into various complex image augmentation approaches such as using GANs to generate images [Al-Dhabyani *et al.*, 2019], enlarging the dataset used for training which naturally improves the performance. The study also found that traditional transformations managed to improve model performance. Many other studies such as [Esteva *et al.*, 2017, Loey *et al.*, 2020] confirmed that minimal transformations such as flipping the images achieved higher prediction accuracy in their application.

We also found that transfer learning architectures are crucial for yielding reliable results as any given patient's dataset is likely too small to construct a CNN architecture. This study by Sun *et al.* [2020] built a CNN using DenseNet models for the prediction of breast cancer masses and achieved a test accuracy of 91%. Zhang *et al.* [2020] also studying the diagnosis of breast cancer tumours in ultrasound images found that when compared with other architectures such as VGG16, ResNet50, and VGG19, InceptionV3 performs the best. Prediction competitions on Kaggle have proven successful with architectures such as VGG16 and Resnet, predicting masses in ultrasound images with 70-80% accuracy [Kaggle, 2016].

For the object detection piece, along with the YOLO framework, we considered RCNNs but research showed YOLO performs well in Kaggle competitions [Kaggle, 2021] and is generally faster than RCNNs. As such, we went with the YOLO model [Gandhi, 2018].

4. Data Science Methods

The general data science workflow for this project is shown in Fig. 3. We discuss each step of the workflow in more detail in the sub-sections below.

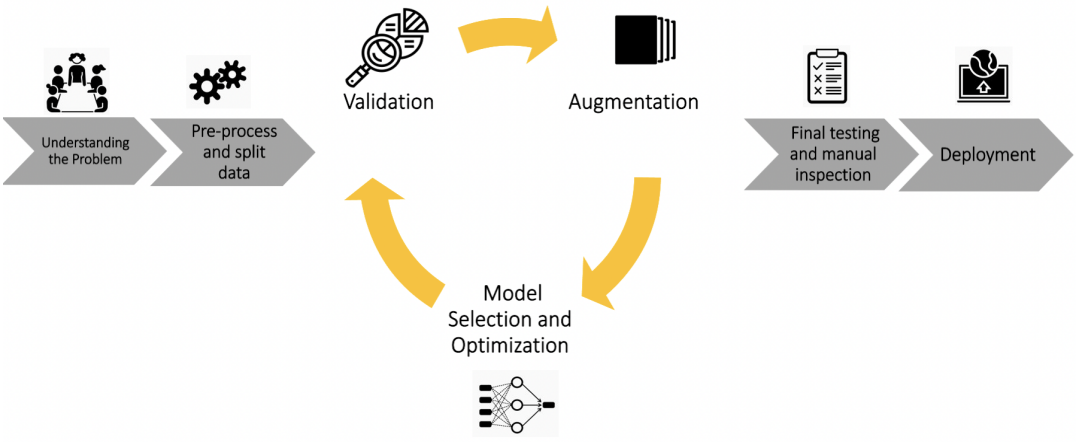

Fig. 3 General data science workflow for this capstone project.

4.1 Data Preparation

4.1.1 Data Splitting and Baseline Model

Before beginning any model training, the data was split into train, validation and test splits of 70%, 15% and 15% respectively. Here, we define baseline as a model that only predicts negative and has 62% accuracy. This accuracy corresponds to the proportion of negatives in the dataset. Our goal for this project was to develop a model that performed better than baseline.

4.1.2 Image Augmentation

Given the small size of the dataset, image augmentation techniques were used to expand the size of the training set and improve the model's generalizability. We explored the pytorch [Paszke *et al.*, 2019] and the albumentations libraries as they had proven successful in increasing accuracy in previous work [Al-Dhabyani *et al.*, 2019]. We also attempted to leverage a machine learning tool called autoalbument from the latter library that searches for and selects the best transformations. However, following several runs, the accuracy did not improve from baseline. We suspect it may be that the transformations were too complex for the model to learn anything significant. Using simpler transformations to augment the data was complemented by results found in our literature review [Esteva *et al.*, 2017, Loey *et al.*, 2020]. A variety of classic transformations were tested and the model's performance on these augmented datasets were documented here. The transformations that led to the best performance were adding random vertical and horizontal flipping along with random brightness and contrast adjustment, with a probability of 50%, augmenting the images as seen below in Fig. 4.

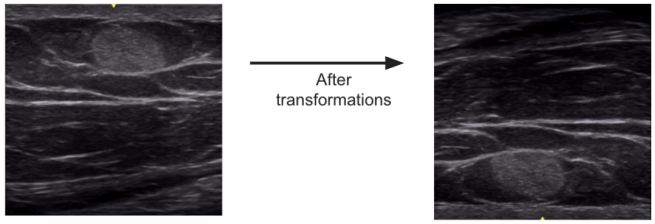

Fig. 4 Final image transformations included random vertical and horizontal flipping and random brightness and contrast adjustment.

4.2 Model Development

The augmented data is then used to train a CNN model using transfer learning, a technique utilizing pre-trained models on thousands of images, which allows for training with our comparatively smaller dataset. Based on our literature review, the transfer learning architectures we chose to investigate were the following: VGG16, ResNet50, DenseNet169 and InceptionV3. Each of the models were incorporated with our small dataset, trained in separate experiments utilizing techniques to optimize the parameters of the model to maximize its ability to learn. To compare the performance of the different architectures, the team considered the accuracy and recall scores of the models when tested on our test set. When comparing the relative scores of accuracy shown in Fig. 5 and recall shown in Fig. 6, DenseNet outperformed the other architectures. DenseNet has also proved successful in similar deep learning applications using small datasets [Zhang *et al.*, 2020], which we suspect is due to its ability to reduce the parameters in a model.

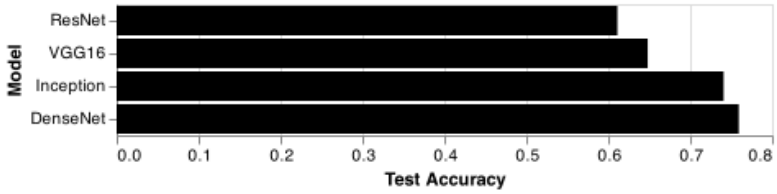

Fig. 5 All four models were tested on a holdout sample to produce these accuracy results.

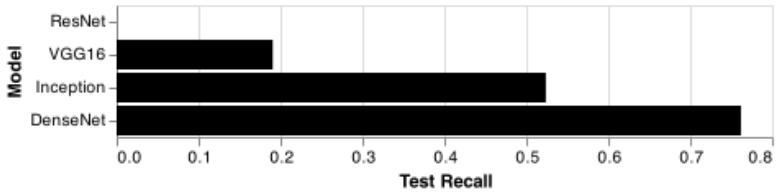

Fig. 6 All four models were tested on a holdout sample to produce these recall results.

Furthermore, we conducted a manual analysis of tricky negative and positive images to evaluate the models' performance using an interactive interface. This application, demonstrated below in Fig. 7, allowed us to compare how confident the models were when misclassifying images and showcased the strengths of the DenseNet architecture.

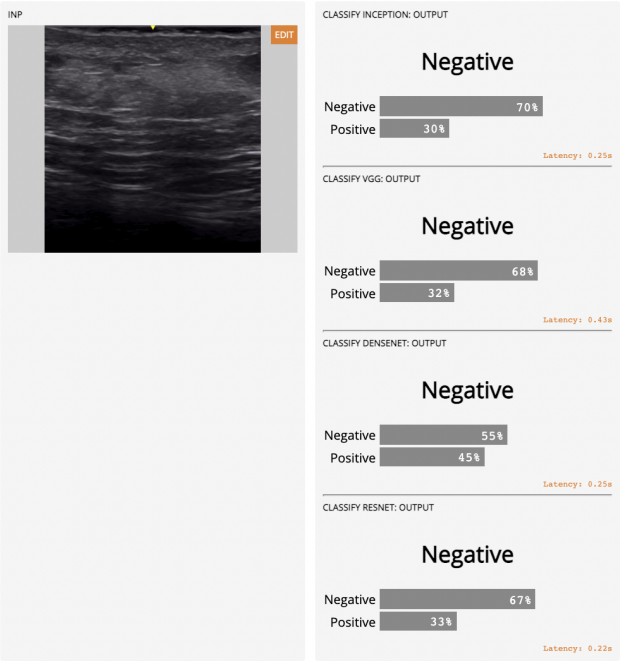

Fig. 7 A manual inspection of tricky examples was conducted. Above, we have a true positive and although, all model are struggling, DenseNet is the least confident in its wrong prediction.

In an effort to make the model more generalizable, we considered various techniques such as implementing dropout layers in our CNN structure, batch normalization and taking an ensemble approach. Furthermore, when identifying liphypertrophic masses, it is more detrimental for a true positive (areas where lipohypertrophy is present) to be falsely labelled as a negative. This is formally known as a false negative and thus, we also attempted to optimize recall, a score where a higher value indicates fewer false negatives. To find out more about these explorations and their outcomes, see Appendix A - Documentation.

Finally, the choice of our optimal model as DenseNet was further motivated by its relatively small computational size as seen in Fig. 8 below, and its compatibility with our data product, further discussed in section 5. Data Product and Results.

Model	Size (MB)
DenseNet	30.49
ResNet	98.56
Inception	100.82
VGG16	537.16

Fig. 8 Comparison of the file size of the different models revealed that DenseNet was the smallest model.

4.3 Object Detection

Our next objective was to implement object detection into our pipeline, giving us the ability to identify the exact location of lipohypertrophy on unseen test images. To implement object detection using a popular framework called YOLO [Jocher *et al.*, 2020], the team created bounding boxes around the location of the lipohypertrophy masses in the positive training images using the annotated ultrasound images as a guide. Next, using the YOLOv5 framework, the YOLOv5m model was trained for 200 epochs with an image size of 300 and a batch size of 8. The team experimented with different training parameters to find that the aforementioned training parameters produce optimal results. Below is a sample of those results identifying the exact location of lipohypertrophy with great confidence.

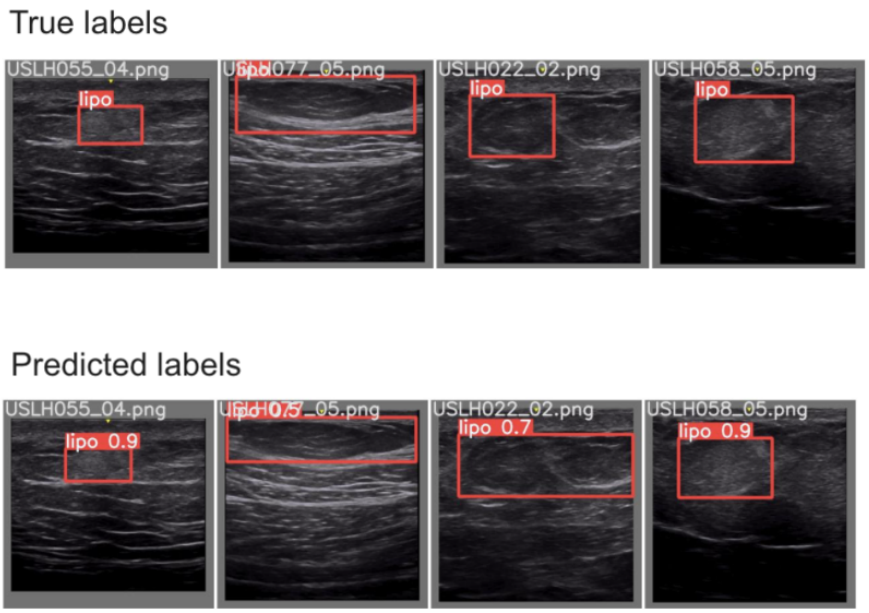

Fig. 9 Our final object detection model results on a test sample reveals promising results. The top row indicates the true location of lipohypertrophy and the bottom row indicates where the model thinks the lipohypertrophy is. The number on the red box indicates the model's confidence.

5. Data Product and Results

Our project comprised of two key data products:

1. Source Code
2. Web application

We describe these products in detail below.

The first deliverable data product is the source code including an environmental file (.yaml), command-line enabled python scripts to perform train-validation-test split on image folders, modeling, evaluation, and deployment, and a Makefile that automates the entire process. The team expects our partner to use the source code to refit the lipohypertrophy classification model as more ultrasound images become available. The python scripts are built with optional arguments for model hyperparameters, which makes this a flexible tool to work with. The Makefile renders the process automatic and makes the model refitting effortless. However, the source code requires a certain degree of python and command line interface knowledge.

The second deliverable is a web application as shown in Fig. 10 for our partner to interact with the lipohypertrophy classification and object detection models. A web application was an ideal choice compared to a local desktop app since the former does not require internal IT clearance, making it more ideal for our partner. The team expects our partner to use this app to upload one or multiple cropped ultrasound images, and get an instant prediction of lipohypertrophy presence along with the prediction confidence. If the prediction is positive, a red box will appear on the image to indicate the proposed lipohypertrophy area. The app is deployed on Streamlit share and Heroku, two free-service deployment platforms. The team sent a special request form to Streamlit to increase the allocated resources under Streamlit's "good for the world" projects program. The request was approved and the app deployed on Streamlit share runs much faster in comparison to that deployed on Heroku; which now serves as a backup. Future (larger) models could be hosted on cloud based servers since they are more flexible and secure.

Fig. 10 Our web application on Streamlit can take multiple images and provides a final prediction and its confidence for a given image. If the model is considered positive, the model will also propose the location of the lipohypetrophy.

6. Conclusions and recommendations

In this project, we aimed to investigate whether supervised machine learning techniques could be leveraged to detect the presence of lipohypertrophy in ultrasound images. Our trained models have demonstrated that this is indeed a possibility as they are accurately predicting the presence of lipohypertrophy on 76% of previously unseen data. Furthermore, our team has developed two data products. The data products are a well-documented source code and an automated machine learning pipeline and an interactive web application.

The open-source licensed source code will allow future researchers and developers to borrow from and build upon this work. The Makefile included with the project makes it seamless to update the model with an expanded dataset. The web application allows healthcare providers to easily interact with our machine learning model to discover which sites are safe for insulin injection and which sites should be avoided.

Although our project has demonstrated that machine learning can be used to detect lipohypertrophy, there are some key limitations that should be addressed before it is used in the clinical setting. These key limitations are as follows:

1. Scarcity of data - We had merely 353 images to develop our model which is a small dataset in the realm of deep learning. The scarcity of our dataset caused variability in different runs of the experiments conducted. We have also noticed that our model's performance is sensitive to data splitting (i.e. which images are in the training, validation and test folders).
2. Limited resources - Limited computing power has been a recurrent challenge in this project and has made it difficult to experiment with tuning all the parameters of our model concurrently.
3. Lack of oversight - We believe that there should be an auditing process involved to ensure that our machine learning model does not propagate any biases that could cause harm to specific patient populations. It would also be useful to have other independent radiologists label the dataset in order to draw comparisons between physicians' and algorithms' performance.

To address the limitations outlined above, our team has made the following recommendations:

1. Addition of more labelled images – in line with the size of the dataset in similar studies [Cheng and Malhi, 2017, Xiao *et al.*, 2018], we recommend increasing the dataset to at least 1000 images and ideally up to several thousand images.
2. Increasing computational resources - we recommend obtaining funding for additional computing resources for both the development and deployment of future models.
3. Adding additional functionality such as a cropping tool within the web interface.

Bibliography

[ADGKA19](1,2)
Walid Al-Dhabyani, Mohammed Gomaa, Hussien Khaled, and Fahmy Aly. Deep learning approaches for data augmentation and classification of breast masses using ultrasound images. *Int. J. Adv. Comput. Sci. Appl*, 10(5):1–11, 2019.

[CM17]

Phillip M Cheng and Harshawn S Malhi. Transfer learning with convolutional neural networks for classification of abdominal ultrasound images. *Journal of digital imaging*, 30(2):234–243, 2017.

[CCL+19]
Jui-Ying Chiao, Kuan-Yung Chen, Ken Ying-Kai Liao, Po-Hsin Hsieh, Geoffrey Zhang, and Tzung-Chi Huang. Detection and classification the breast tumors using mask r-cnn on sonograms. *Medicine*, 2019.

[EKN+17] (1,2)
Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature (London)*, 542(7639):115–118, 2017.

[Gan18]
Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo - object detection algorithms. Jul 2018. URL: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[HZL18]
Qinghua Huang, Fan Zhang, and Xuelong Li. Machine learning in ultrasound computer-aided diagnostic systems: a survey. *BioMed research international*, 2018.

[JSB+20]
missing booktitle in glenn_jocher_2020_4154370

[Kag16]
Kaggle. Ultrasound nerve segmentation. 2016. URL: https://www.kaggle.com/c/ultrasound-nerve-segmentation/overview.

[Kag21]
Kaggle. Vinbigdata chest x-ray abnormalities detection. 2021. URL: https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection/code?competitionId=24800&sortBy=voteCount.

[KPCM18] (1,2,3)
JE Kapeluto, BW Paty, SD Chang, and GS Meneilly. Ultrasound detection of insulin-induced lipohypertrophy in type 1 and type 2 diabetes. *Diabetic Medicine*, 35(10):1383–1390, 2018.

[LMK20] (1,2)
Mohamed Loey, Gunasekaran Manogaran, and Nour Eldeen M Khalifa. A deep transfer learning model with classical data augmentation and cgan to detect covid-19 from chest ct radiography digital images. *Neural Computing and Applications*, pages 1–13, 2020.

[Mad21]
Ken Madden. Machine learning approaches to: 1. diagnosing lipohypertrophy at the bedside, and 2. falls prediction in long term care. 2021. URL: https://github.ubc.ca/MDS-2020-21/DSCI_591_capstone-proj_students/blob/master/proposals/md/Machine_Learning_Approaches_to_Diagnosing_Lipohypertrophy_and_Predicting_Falls.md.

[PGM+19]
Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[SLZ+20]
Qiuchang Sun, Xiaona Lin, Yuanshen Zhao, Ling Li, Kai Yan, Dong Liang, Desheng Sun, and Zhi-Cheng Li. Deep learning vs. radiomics for predicting axillary lymph node metastasis of breast cancer using ultrasound images: don't forget the peritumoral region. *Frontiers in oncology*, 10:53, 2020.

[XLL+18]
Ting Xiao, Lei Liu, Kai Li, Wenjian Qin, Shaode Yu, and Zhicheng Li. Comparison of transferred deep neural networks in ultrasonic breast masses discrimination. *BioMed research international*, 2018.

[YAAK18]

Koichiro Yasaka, Hiroyuki Akai, Osamu Abe, and Shigeru Kiryu. Deep learning with convolutional neural network for differentiation of liver masses at dynamic contrast-enhanced ct: a preliminary study. *Radiology*, 286(3):887–896, 2018.

[ZHC+20](1,2)
Heqing Zhang, Lin Han, Ke Chen, Yulan Peng, and Jiangli Lin. Diagnostic efficiency of the breast ultrasound computer-aided prediction model based on convolutional neural network in breast cancer. *Journal of Digital Imaging*, 33:1218–1223, 2020.

Appendix A - Documentation

In this section of the Appendix, we attempt to highlight at a high level, some of the technical decisions that were taken throughout the duration of this capstone project. This section presents some of the technical details behind those decisions.

Data Augmentation

The team initially used a web interface provided by the albumentations library, powered by PyTorch for image transformations, that allowed us to see what the different transformations would look like. Based on that, we chose some transformations to test in our pipeline. This experiment can be found here. The final model was chosen based on the transformations yielding the highest accuracy score.

To use the albumentations library:

- This guide is extremely helpful for our classification task. You develop your own class that inherits from the torch dataset. To see an example of this implementation for our dataset, please check this notebook.

We also tried a machine learning tool called autoalbument which allows for the selection of optimal transformations.

Loss Functions

A loss function, also known as cost function or error function, is a function that maps a set of parameter values for the network onto a scalar value. The scalar value is an indication of how well the parameters are completing their assigned tasks. In optimization problems, we seek to minimize the loss function.

The following functions were considered for the choice of the loss function used in the convolutional neural network model:

1. BCEWithLogitsLoss
2. NLLLoss
3. Sum of Log Loss
4. Hinge
5. The mean absolute error

Loss Function Implemeted in CNN Model:

- BCEWithLogitsLoss: Cross-Entropy loss (nn.BCEWithLogitsLoss) combines a Sigmoid layer and the BCELoss in one single class. This loss function in often used in classification problems as a measure of reconstruction error. It uses log loss, which for logistic regression is a special case for cross-entropy loss for multi-class classification.

Alternative Loss Functions Considered:

- NLLLoss: Negative Log-Likelihood Loss This function expands the binary cross-entropy loss. It requires an additional logSoftMax layer in the last layer of the network. As it outputs probablities, they must sum to 1. The function also has various reduction of output options such as the weighted mean of output or sum.
- Sum of Log Loss: This function allows for the gradient to update with more incentive
- Hinge: This function, nn.HingeEmbeddingLoss, is used more for measuring similarities. It tries to maximize the margin between decision boundary and data points. The main advantage is that the function penalizes incorrect predictions a lot but also correct ones that aren't confident (less). That is, confident correct predictions are not penalized at all.
- L1 MAE: The mean absolute error

Structure of CNN Model

Based on our Liturature Review, the following transfer learning architectures were considered:

1. DenseNet
2. Inception
3. VGG
4. ResNet

The model variants that were considered are documented below, along with some notes thought to be relevant.

- VGG16: It makes the improvement over AlexNet by replacing large kernel-sized filters (**11** and **5** in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another.
- DenseNet121: DenseNet121 proved to be the best performing model given the Lipohypertrophy data. To see this exploration, click here.
- ResNet50
- InceptionV3

Batch Normalization

We ensured all models have batch normalization implemented. This will standardize the inputs of a network improving the overall efficiency by reducing the number of epochs required to train for a given model.

Dropout Layer

In an effort to reduce the generalization error and high confidence in misclassifications, we considered the implementation of dropout layers within the model's architecture. Dropout layers will randomly drop nodes in the network such that the model learns more robustly and the validation performance is improved. A dropout rate can be specified to indicate the probablity at which nodes are dropped. The dropout layers experiments with varying dropout rates did not show any success for our dataset. Although the dropout layers managed to reduce misclassifications in the models, the overall reduced accuracy was not remarkable enough to implement those layers in our optimal model. To see this exploration, click here.

Ensemble Model

Another way to reduce the generalization error is to average the predictions from all four models, called an ensemble. However, an ensemble would not be feasible as it requires lots of resources such as enough CPU to train the models and in our case, since the models performed too similarily, the average accuracy would be lower.

Recall Optimization

To reduce the generalization error, we considered varying the pos_weight argument in the loss function. Positive weight (pos_weight) is one argument of the loss function used in our CNN model which, if greater than 1, prioritizes the positive examples more such that there is a heavier penalization (loss) on labelling the positive lipohypertrophy examples incorrectly. Increasing the positive weights to more than 1, we would expect the recall score to improve since there would be heavier penalization (loss) on positive examples, in an attempt to reduce false negatives (a true positive where the model predicts is negative). However, this method was not implemented in our final model as it proved to be unstable. After conducting a few experiments, we found high variance in the results.

Hyperparameter Optimization

Bayesian Optimization with Ax

The general idea with the Bayesian method is that not every possible parameterization in the defined parameter space is explored. The process tunes parameters in a few iterations by building a surrogate model. A surrogate model is a probablistic model that uses an acquisition function to direct the next sample to make up configurations where an improvement over the current best parameterization is likely. Bayesian Optimization with Ax, a recently developed package for optimization by Facebook, relies on information from previous trials to propose better hyperparameters in the next evaluation. The Gaussian process is as follows:

1. Build “smooth” surrogate model using Gaussian processes (initially Gaussian with mean 0 and variance equal to the noise in data. The surrogate model updates to a Gaussian model with mean equal to the estimated mean and updated variance based on the previous trial.
2. use the surrogate model to **predict** the next parameterization (estimated mean), out of the remaining ones and quantify uncertainty (updated estimated variance)
3. combine predictions and estimates to derive an acquisition function and then optimize it to find the best configuration.
4. Observe outcomes

Fit new surrogate model and repeat process.

As a way to guide the surrogate model in the prediction of the next parameter configuration to explore, aquisition functions are utilizied.

- Three common examples include:

1. Probability of Improvement (PI).
2. Expected Improvement (EI).
3. Lower Confidence Bound (LCB)

The most common technique used in Bayesian Optimization is the second choice: Expected Improvement. As such, that was used for our experiments found here.

An Ax tutorial can be found here.

The main advantages to Bayesian Optimization are:

- its ability to find better parameterizations with fewer iterations than grid search and;
- striking a balance between exploration and exploitation where exploration refers to trying out parameterization with high uncertainty in the outcome and exploitation refers to the surrogate model predicting likely parameterizations.

In addition to our Bayesian experiments, we consulted results in the following papers regarding classical transformations applied to small:

- A Review of Bayesian Optimization
- Bayesian Hyperparameter Optimization with BoTorch, GPyTorch and Ax.

Appendix B - Glossary

CPU: stands for for Central Processing Unit, is a portion of a computer that retrieves and executes instructions. CPU’s are used in this context to execute a sequence of stored instructions (commands).

Ensemble Learning: the use of combining multiple learning algorithms to obtain improved predictive performance than any individual learning algorithm predictive performance. In this context, an ensemble approach could be achieved by averaging the predictions from multiple models.

Epochs: a number defining the number of times that the learning algorithm will pass through the entire training dataset.

Executable: a file or program with the ability to be run by a computer.

False positive: a false positive, also referred to as type I error in Statistics, is an error in prediction where a negative example is misclassified as a positive example. In this context, a false positive is an indication of a Lipohypertrophy mass when there is none in reality.

False negative: a false negative, also referred to as type II error in Statistics, is an error in prediction where a positive example is misclassified as a negative one. In this context, a false negative is an indication of a no Lipohypertrophy presence when there is in reality.

GAN: short for Generative Adversarial Network, GANs are a machine learning model where two neural networks compete against each other. In this context, GANs can be used to develop authentic-looking but fake images of ultrasounds to build a large dataset that can be used for future applications.

Gaussian process: a Gaussian process is a stochastic process, a collection of random variables with a time or space component, that allows finite collection of random variables to have a multivariate normal (bell-shaped) distribution. That is, every finite linear combination of random variables is normally distributed.

RCNNs: short for recurrent correlation neural networks, RCNNs are a machine learning model used for object detection. RCNNs learn the data but dividing it into sections and taking the most important features to then classify.

Radiomics: a technique using data characterization algorithms to extract a large number of features from radiographic medical images such as ultrasound images.

Recall: a ratio representing the number of true positives divided by the number of true positives plus the number of false negatives. Optimizing recall means to reduce the number of false negatives.

Transfer learning: a technique that allows us to leverage pre-existing models that have already been trained on thousands of images from various data sources and applying them to our dataset.

Sigmoid layer: a sigmoid layer in a neural network applies a sigmoid function to the input such that the output is bounded in the interval (0,1).

YOLO: short for You Only Look Once, YOLO is an object detection framework that uses convolutional neural networks to train on images and optimize detection performance in real-time.

By Ela Bandari, Lara Habashy, Javairia Raza, and Peter Yang