Evaluation of the precision and accuracy in the classification of breast histopathology images using the MobileNet v3 model

Team 1: Ken DeVoe and Gary Takahashi

Candidate paper for the degree of M.S. Applied Artificial Intelligence

AAI-501-01-SU23 – Introduction to AI

Shiley-Marcos School of Engineering, University of San Diego

Dr. Ebrahim Tarshizi

August 14, 2023

Evaluation of the precision and accuracy in the classification of breast histopathology images using the MobileNet model

Abstract

Accurate surgical pathologic assessment of breast biopsies is essential to the proper management of breast lesions. Identifying histologic features, such as nuclear pleomorphism, increased mitotic activity, cellular atypia, patterns of architectural disruption, as well as invasion through basement membranes into surrounding stroma and normal structures, including invasion of vascular and lymphatic spaces, help to classify lesions as malignant. This visual assessment is repeated on numerous slides taken at various sections through the resected tumor, each at different magnifications. Computer vision models have been proposed to assist human pathologists in classification tasks such as these. Using a newer and more compact architecture, MobileNetV3, we attempted to classify breast cancer images in the BreaKHist_v1 breast pathology dataset to determine performance of this model out-of-the-box. Using just transfer learning based on ImageNet embeddings without special feature extraction, we were able to correctly classify histopathology images broadly as benign or malignant with 0.945 precision, 0.932 recall, and an F1 score of 0.939. The ability to classify into histologic subcategories was varied, with the greatest success being with classifying ductal carcinoma (F1 score = 0.870), and the lowest success being with phyllodes tumor (F1 score = 0.472). Suggestions to further refine the model are discussed.

Introduction

Breast cancer is the most common cancer afflicting females, second only to skin cancers, and now afflicts nearly one in three women each year (1). This condition usually presents either as a palpable breast mass, or is detected by screening mammography. Once a lesion is identified, a core needle biopsy is performed to assess the nature of the abnormality, and a surgical pathologist is tasked with determining whether the biopsied lesion is benign or malignant. Predicting how the tumor is likely to behave can be determined by evaluating the histologic features such as cellular morphology, the degree and nature of architectural distortion, as well as invasion through basement membranes into normal structures including blood vessels and lymphatics (2). Pathologists typically evaluate numerous slides involving various sections through the tumor, each at different magnifications.

Diagnostic radiologists who interpret mammographic imaging have benefited from computer-assisted identification of suspicious lesions, there has been interest in using deep learning processes to assist the pathologist in histologic classification. Previous attempts have been made to utilize convolutional neural net (CNN) algorithms to classify breast cancer pathologic images into benign and malignant categories, using larger image processing architectures that have been used to successfully classify non-medical, more everyday images. Others have used extensive image preprocessing user feature extraction algorithms to assist in the identification of suitable features for more efficient learning. In this project, we sought to investigate if a modern, and more compact CNN, would have the power to classify breast cancer histopathology images from a labeled dataset. To accelerate the training process, we planned to incorporate transfer learning, using the pretrained embeddings from ImageNet, a large collection of images which was the basis of the competition which was won by the AlexNet architecture (3). The ability to obtain results comparable to that of previous efforts, executed on hardware that is well within the consumer space, and without the need for extensive training, would demonstrate the robustness of the CNN architecture as well as to support efforts to further refine our classification model.

Dataset Selection

BreaKHis_v1 is a large dataset consisting of 9109 microscopic images collected from 82 patients. Of these, 7909 images were available for public downloading (Spanhol, 2016). This dataset includes 2480 images of benign breast lesions, and 5429 images of malignant lesions. Regions of interest (ROI) have been identified by a pathologist and the images were obtained at the indicated magnifications. Benign breast lesions were divided into four categories: adenosis, fibroadenoma, phyllodes tumor and tubular adenoma. The malignant conditions selected for inclusion were ductal carcinoma, lobular carcinoma, mucinous carcinoma and papillary carcinoma. Each histologic category was represented in four magnification levels: 40X, 100X, 200X and 400X. This is based on the objective magnification of the microscope multiplied by the 10X eyepiece. The number of images at each magnification is as follows:

	40X	100X	200X	400X	Total
Adenosis	114	113	111	106	445
Fibroadenoma	253	260	264	237	1015
Phyllodes	109	121	108	115	454
Tubular adenoma	149	150	140	130	570

	40X	100X	200X	400X	Total
Ductal Ca	864	903	896	788	3452
Lobular Ca	156	170	163	137	627
Mucinous Ca	205	222	196	169	793
Papillary Ca	145	142	135	138	561

Table 1: BreaKHis dataset data distribution

We selected the BreaKHis_v1 dataset as the most suitable for our objectives, as alternative breast cancer datasets either were no longer available, contained fewer or smaller images for classification, offered only binary classifications (i.e. benign vs malignant) or were suited for the detection of metastatic spread in the background of lymphoid tissue.

Exploratory Data Analysis

The size of each image in the BreaKHis_v1 dataset is 700 x 460 pixels and is stored in PNG format. Examining random samples from the dataset revealed that the images were of good quality, and suitable for training. Images obtained at low magnitude reveal glandular structures, integrity of the stroma, and architectural distortion by the invasion of neoplasm. Higher magnifications reveal the presence of mitotic figures, nuclear atypia, degree of cellular differentiation and abnormal growth patterns), as well as the presence or absence of invasion through basement membranes (signifying metastatic spread) as well as abnormal tumor invasion through lymphatic or vascular channels. Because each magnification contributes useful information that would be important to classification, we did not restrict the analysis to specific magnifications, but elected to train the model on all magnifications collectively. We considered the option of training on subsets only if our model or hardware exhibited indications of lack of capacity to analyze the entire training dataset.

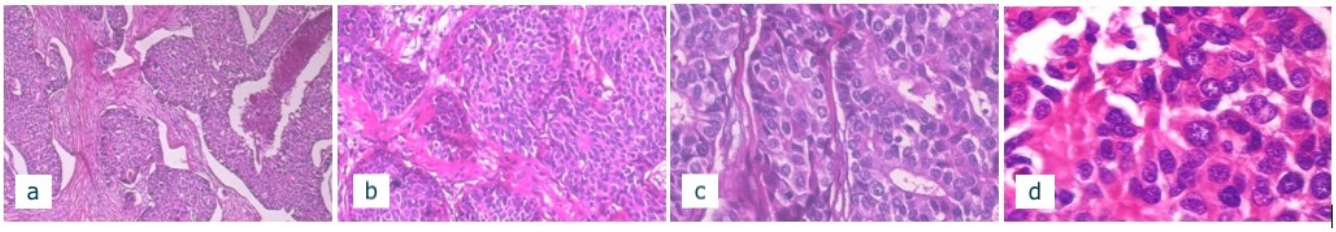


Fig 1. Examples from the BreaKHis_v1 ductal carcinoma dataset, seen at magnifications of a) 40X, b) 100X, c) 200X, d) 400X. Lower magnifications more clearly display architectural details of tumor penetration into normal breast tissue, while at higher magnifications, cellular and nuclear detail become more important. Presence of metastatic cells in the lymphatic channels and capillaries are better appreciated at the higher magnifications.

Equipment and platform

Training of the model was done on Apple desktop computers. One was based on a Mac M2 processor with 64 GB of RAM running macOS 13.5. The other computer was a MacBook Air with an M1 processor and 8 GB of RAM running on macOS 13.4.

Jupyter Lab 4.0.3 was used to load the datasets, define and train the MobileNetV3 model on the breast histology datasets.

Model Selection

Eleven years have elapsed since AlexNet debuted in the ImageNet competition, and since then, CNNs have been refined, often incorporating more convolution and pooling modules, improving on local response normalization, introducing residual networks, and inception networks. The

MobileNet family is built on the concept of depthwise-separable convolutions, forcing the model to learn rather than memorize, which would promote overfitting, while leading to lower computational cost at deployment (4). MobileNetV3 built on MobileNetV2's improvements, by introducing the "squeeze and excite" algorithm in parallel to all components of the residual block; using Neural Architectural Search to improve accuracy (hence the title of the V3 paper - Howard, 2019); and replacing some of the sigmoid activations with the "swish" function or with its piece-wise linear hard analog, to improve performance (since sigmoid activations are more expensive to compute on mobile devices) but without compromising on accuracy. This enabled some of the last stages to be shortened also without decreasing accuracy. MobileNetV3 is divided into Small and Large versions, with the Small version to be run on reduced-resource devices.

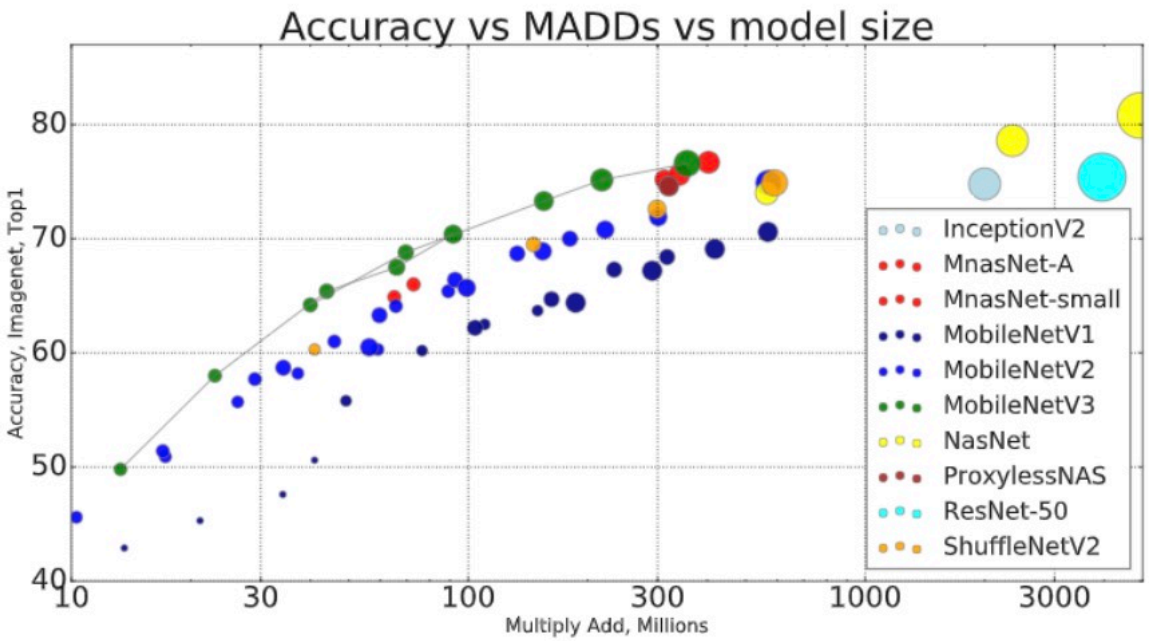


Fig 1. Comparison of ImageNet performance of different architectures run on the Google Pixel 4 phone. y-axis is accuracy. x-axis is number of MADD (multiply-add accumulates) units. Higher MADD values indicate lower efficiency. (Howard, 2019)

The goal of our investigation was to determine if the MobileNetV3 Large would run on hardware available to us (consumer Apple computers), using the ImageNet-pretrained embeddings, to classify histopathology images into the labeled categories with precision and accuracy comparable to what has been previously published.

MobileNetV3 is publicly available using `tf.keras.applications.MobileNetV3Large()`.

Data Augmentation

Data augmentation is performed when the sample number in the dataset is limited, and is considered insufficient for adequate training. Augmentation may also be considered when the dataset is contained an imbalance of one or more target sets, since in an effort to minimize loss, the model will err on the side of selecting the most represented class. The BreaKHist dataset is large, and our concerns were that the size of the dataset might prove to be more than the model could reasonably handle. Once we were assured that the model would train on the large dataset, then we could turn our attention to data augmentation. In this study a simple data augmentation strategy of randomly flipping the input image horizontally was applied. Future work could include vertical flips of the image as well as rotations of the input image. Another possible strategy to provide a more even balance to the number of images in each training subset would be to implement image augmentation to the under-represented subsets. Time constraints have led us to defer this tactic in the interest of completing the main objective.

Pre-processing

MobileNetV3 expects a target image size of 224 x 224 px with 3 channels for RGB color representation, a size that has been adopted for reasons of tradition, based on the original AlexNet model. Images were preprocessed using `tensorflow.keras.applications.mobilenet_v3.preprocess_input`.

Images in the BreaKHis dataset have a native resolution of 400 x 260 px, however. We imported the image files using `tensorflow.keras.utils.image_dataset_from_directory()`. This function performs automatic resizing. We left the `crop_to_aspect_ratio` parameter to the default setting, acknowledging that there would be some distortion, so that the software would not crop images indiscriminately, potentially eliminating key histologic features from being available for training.

Feature extraction has been utilized in other efforts as a means of strengthening features, such as contrast enhancement to emphasize edges, or semantic segmentation to highlight regions of interest (ROI). Implementation of this step has required input from trained pathologists to confirm correct identification of ROI, and in the setting of metastatic breast cancer (Campanella, 2019). We sought to determine the performance of our model without such human input, to classify breast pathology into four benign and four malignant categories, with the background of normal breast tissue, a conceptually more difficult problem.

Model Architecture

The MobileNetV3 Large architecture was selected, as it offered slightly improved accuracy. The model comes pre-trained with ImageNet embeddings, and we elected to take advantage of transfer

learning. Outputs from MobileNetV3 were flattened to a 1 x 47040 vector, then passed to three Dense layers, followed by a Softmax layer with 8 outputs. Dropout and BatchNormalization were used to reduce overfitting. As shown in figure 2 the final model consisted of 6.0 million parameters with 50.0 % coming from MobileNetV3 Large and the remaining 50.0 % coming from additional layers added to the end of the model.

Layer (type)	Output Shape	Param #
input_211 (InputLayer)	[(None, 224, 224, 3)]	0
sequential_35 (Sequential)	(None, 224, 224, 3)	0
MobilenetV3large (Functional)	(None, 7, 7, 960)	2996352
flatten_103 (Flatten)	(None, 47040)	0
dropout_103 (Dropout)	(None, 47040)	0
dense_408 (Dense)	(None, 64)	3010624
batch_normalization_102 (BatchNormalization)	(None, 64)	256
dense_409 (Dense)	(None, 32)	2080
dense_410 (Dense)	(None, 16)	528
dense_411 (Dense)	(None, 8)	136

Total params: 6009976 (22.93 MB)

Fig 2. Architecture of model used for image classification.

Results

Model training and hyper-parameter tuning were focused into three key areas; data augmentation, layer selection for training and learning rate selection.

Data Augmentation

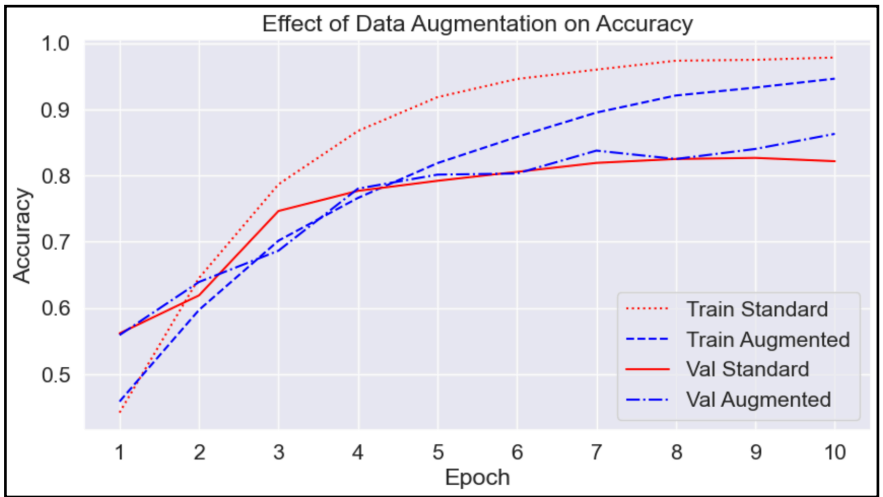


Figure 3 shows the training and validation results from employing simple random horizontal flip data augmentation.

Fig 3. Comparison of model training and validation accuracy with and without using data augmentation. Both training runs use the same hyper parameters with a learning rate of 0.001 and learning rate decay of 0.95 per epoch.

Overall data augmentation has a significant positive effect on the training of the model. Starting around epoch 3 it can be seen that the standard model with no data augmentation starts to overfit with the training accuracy outpacing the validation accuracy. After 10 epochs the gap from training accuracy to validation accuracy is large at 98% compared to 82%. When data augmentation is introduced overfitting does not start until epoch 6 and at epoch 10 training and validation results are much closer at 95% compared to 86%. Due to the net positive result of reducing overfitting and improving validation accuracy data augmentation is applied to model training in all of the following results.

Layerwise Learning

During model training one of the key parameters to select was where to freeze and where to start training the MobileNetV3 model. In general if not enough of the model is fine tuned then it may perform poorly on tumor images, whereas if too much of the model is retrained then useful information from pre-training on ImageNet will be lost. This effect can be seen in Figure 4. Starting training at the lower layers in the model up to around layer 130 leads to poor, sporadic performance of the overall model. This is likely due to pre-training information loss where the model loses the basic ability to recognize image features. After approximately layer 150 performance also shows a slight decline likely due to not enough fine tuning towards tumor images. Based on these results layer 150 was selected to as the optimal layer to start training the model.

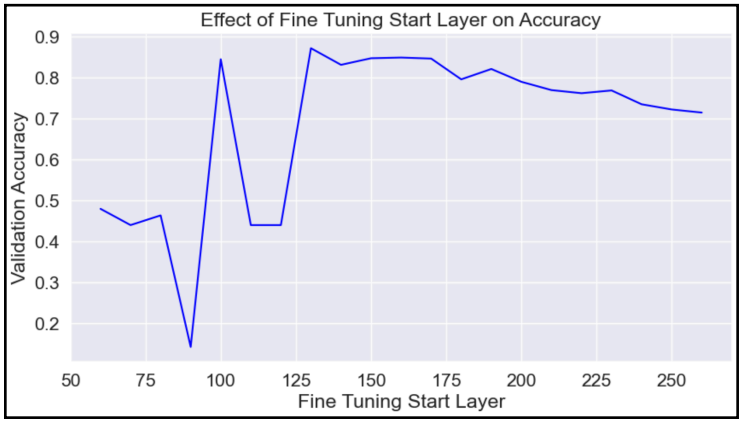


Fig 4. Validation accuracy of model after 10 epochs based on the layer fine tuning was started at. Each training session used an initial learning rate of 0.001 and epoch decay rate of 0.95 . This graph represents 21 unique model training runs. For reference the model includes 268 layers total with 263 coming from MobileNetV3.

Learning Rate Selection

Learning rate was selected as a combination of two factors; the initial learning rate and the rate of decay per epoch. Figure 5 shows the model validation accuracy as a function of initial learning rate and epoch decay rate. All models were trained for 10 epochs and fine-tuning was started at layer 150 out of 268.

A high starting learning rate of 0.01 gives relatively poor results compared to both 0.001 and 0.0001 . This is likely due to both resetting pre-trained weights and overshooting the optimal solution . Training starts at run 150 which results in 113 layers of 263 being re-trained for MobileNetV3. Using a high learning rate for a significant amount of layers in the pre-trained model will effectively remove learnings the model gained from ImageNet. In addition high learning rates are known to overshoot optimal solutions (Russel & Norvig, 2020). Of the three learning rates chosen 0.001 gives the best overall results as 0.0001 appears to learn too slowly to achieve optimal accuracy.

For the best performing learning rate starting values an epoch decay rate of 0.95 gives the best overall result. A decay of 1.0 is effectively no decay, and likely leads to the model overshooting the optimal solution at later epochs. More aggressive epoch decay rates of 0.9 and 0.8 likely slow learning too much at later epochs. For final analysis a start learning rate of 0.001 and epoch decay of 0.95 was chosen.

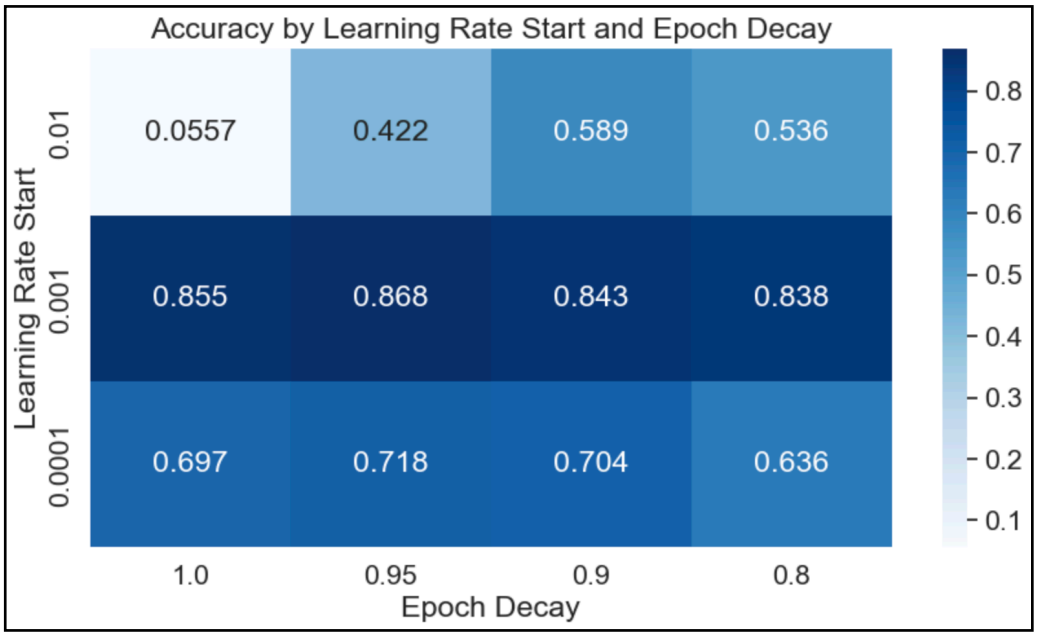


Fig 5. Validation accuracy of model based on the initial learning rate and epoch decay rate. Fine-tuning was started at layer 150 out of 268 for all models and each model was trained for 10 epochs.

Final Results

With 10 epochs of training, the model was able to classify the histology slides as benign vs malignant with recall of 0.968, a precision of 0.987 and F1 score of 0.977 (Fig 2A). Amongst the subtypes of each major category, the model was able to properly classify each histology to varying degrees. The model had greatest success in identifying ductal carcinoma, adenosis, and tubular adenoma, with accuracies of ≥ 0.9. Mucinous carcinoma and phyllodes tumors were accurately identified in 0.85 of cases. Accurate classification for papillary carcinoma was 0.74 and lobular carcinoma was 0.56.

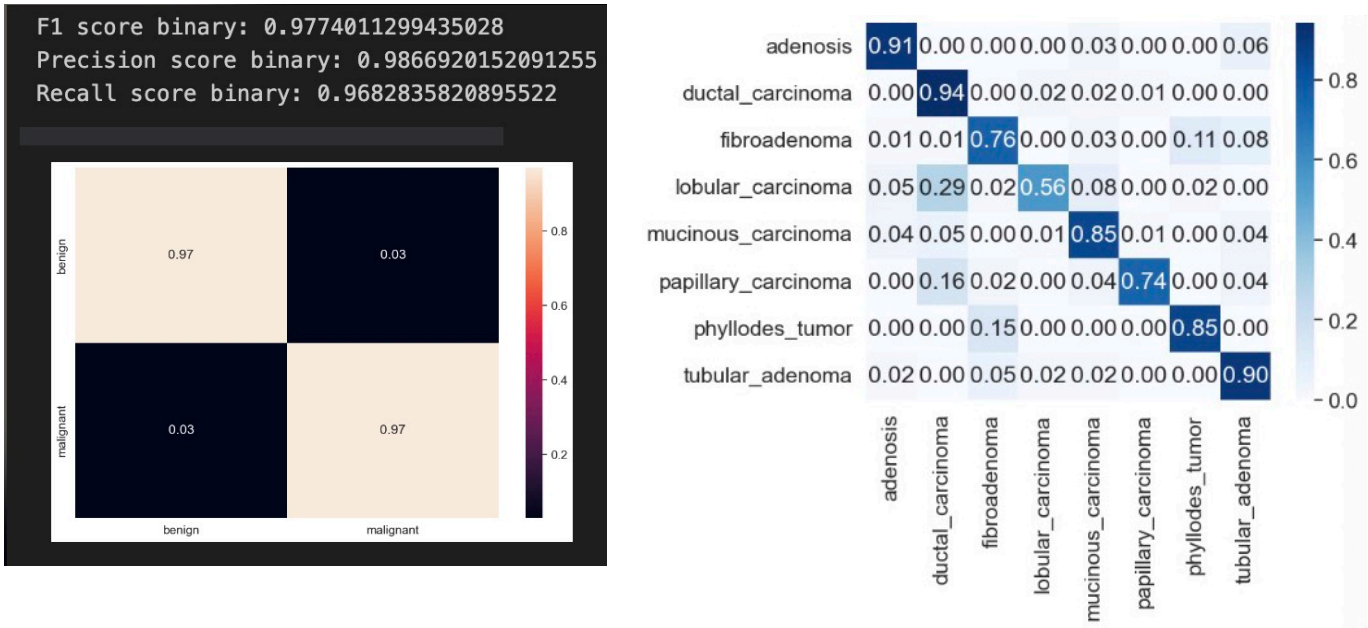


Fig 2. A) Confusion matrix generated for the binary classification of benign vs malignant.
B) Confusion matrix for accuracy in classifying histologic subtypes.

Incorrectly classified images

To gain insight into how the model erred in classifying some of the histopathology images, we identified the misclassified images as well as their predicted labels (ŷ) and the ground truth label. A few of these images are presented here.

As can be seen in Fig 7, the image on the top left that was misclassified as lobular carcinoma contained mainly tissue stroma or possibly tissue necrosis, with very few identifiable cells. The image on the top right, misclassified as phyllodes tumor was reasonable in quality, however in this selected image it may be difficult for even a human pathologist to make the distinction between fibroadenoma and phyllodes tumor. The image at the bottom left, also misclassified as phyllodes

tumor, is of such high magnification that the image consists of a sheet of nuclei with dispersed chromatin and indistinct cellular borders, and the possibilities of its provenance are numerous. The final image in the lower right, misclassified as ductal carcinoma, consists mainly of cellular outlines, and may possibly represent adipose tissue, with little material that can be identifiable as carcinoma. It is therefore reasonable that images like this were misclassified, as they would pose a challenge for even expert human evaluation (approaching Bayes optimal error rate).

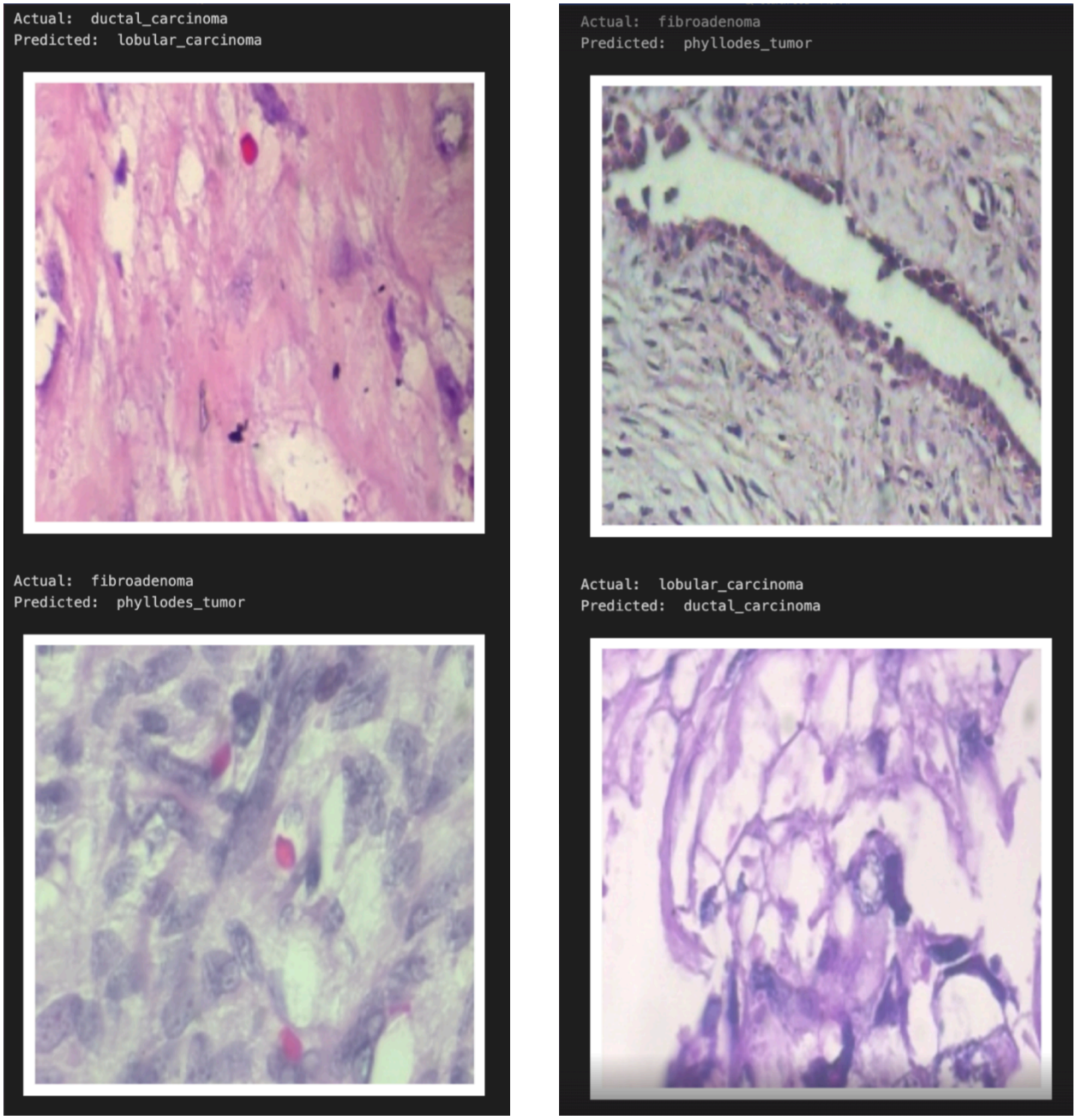


Fig 7. A sample of the images that the model misclassified.

Discussion

We have shown that a convolution architecture, designed for increased accuracy, decreased latency, and decreased resource utilization, was able to properly categorize breast cancer histopathology images as to being benign vs malignant with >96% precision and recall. The model had some success in classifying the images as to histologic subtypes. Accuracy was best with the classification of ductal carcinoma, which is important since it is the most common subtype of breast cancer, representing 70% to 80% of all breast cancers (Li, 2003). This may account for the increased proportion of images representing this histology in the dataset. This may account for the relative

accuracy in classifying ductal carcinoma, as there were more examples for the model to train on. As the model trains to decrease loss, the increased prevalence of this category likely led to more rapid reduction in loss function values than for the other categories. Nevertheless, confident identification of infiltrating ductal carcinoma would be of value, as it could potentially assist in the confirmation of this histology in over half of breast cancer cases.

To put our findings into context, we will review some of the work that inspired our own efforts. Spanhol et al (2016) used six feature extractors and four different unsupervised classifiers on the BreaKHis dataset, at each of the magnifications available. Their model best classified malignant histology correctly, with accuracies of around 0.94, but the accuracy of classifying benign tissue ranged from 0.38 to 0.75 at the various magnifications. Error was most noted in classifying fibroadenomas, which constituted around 30% of errors at every magnification. The ROC AUC was around 0.8 at all magnifications, and they did not report results with all magnifications combined. As stated above, pathologists extract architectural information at low magnification and nuclear and cellular detail and tissue invasiveness at high magnification, so constraining the model to classify at one magnification level is an unnecessary restriction.

Zhu et al (2019) trained a custom CNN model with ideas taken from the Inception architecture (residual network connections), but with the "squeeze-excite" features of MobileNetV3. Their model was trained on the BreaKHis and BACH breast cancer datasets. They also reported accuracy based on each magnification class, and achieved greater than 0.9 AUC on ROC analysis, but this was based on "channel pruning" of the model, ostensibly to decrease computing burden, although it raises some concern about how this model would generalized to other datasets.

Araújo, et al (2017) studied an unspecified CNN (but which resembles AlexNet) together with a SVM classifier, and trained on the Bioimaging 2015 challenge dataset, consisting normal as well as benign, and malignant breast cancer, some invasive and some *in situ*. "Patchwise accuracy" was 0.667 to 0.776. Image-wise accuracy was 0.778 to 0.833, using a voting system, the process of which is not well-described.

Joshi et al (2023) trained an Xception model on the BreaKHis and IDC breast cancer datasets. The objective was the classify the material into benign and malignant categories. ROC AUC was 0.921 on the BreakHis dataset and 0.881 on the IDC dataset. Accuracy data regarding further subclassification into histologic subsets was not presented.

Amerikanos, et al (2022) used feature augmentation with semantic segmentation to train Facebook AI Research's Detectron2 architecture on the TUPAC16 challenge dataset. The investigators sought to avoid having a human pathologist determine the segmentation, and used an AlexNet trained with parameters published previously (Janowcyzk, 2016). The objective of this study, however, was to automatic the identification of nuclear to facilitate feature selection, and not histological classification.

We recognize that in the real world, there are more histological possibilities than are represented in this limited dataset, such as triple-negative breast cancers and mixed-histologies. Indeed we coded a predictor that performed well on selected random images from the dataset, but performed less accurately on pathology slides obtained from the Internet. One reason for this is that even within the individual tissue types, there exist substantial variability in appearance between low-grade malignancies and high-grade malignancies. Details of this kind were not provided with the dataset, and all examples of one tumor subtype were placed into the same diagnosis bin. An informal observation was the model seemed to predict better with images that were closer in dimension to that provided in the dataset. Larger images used for prediction, even though they contained more detail, appeared to be less predictable to the model, suggesting that pre-processing of larger images may have introduced noise that reduces prediction accuracy. Accuracy may also be improved if external material were provided to our model in varying magnifications to exploit all the training gained in our model. Furthermore, surgical pathologists do not rely solely on visual imaging of hematoxylin-eosin stained slides, but also on immuno-histochemical staining for diagnostic confirmation.

One of the known limitations of deep learning in convolutional architectures is that the trained weight embeddings are notoriously in a "black box". Image data that is passed through CNNs undergo drastic transformations through convolution, pooling, flattening, such that examination of layers deep within the model reveal no discernible relationship to the original input (Fig 3).

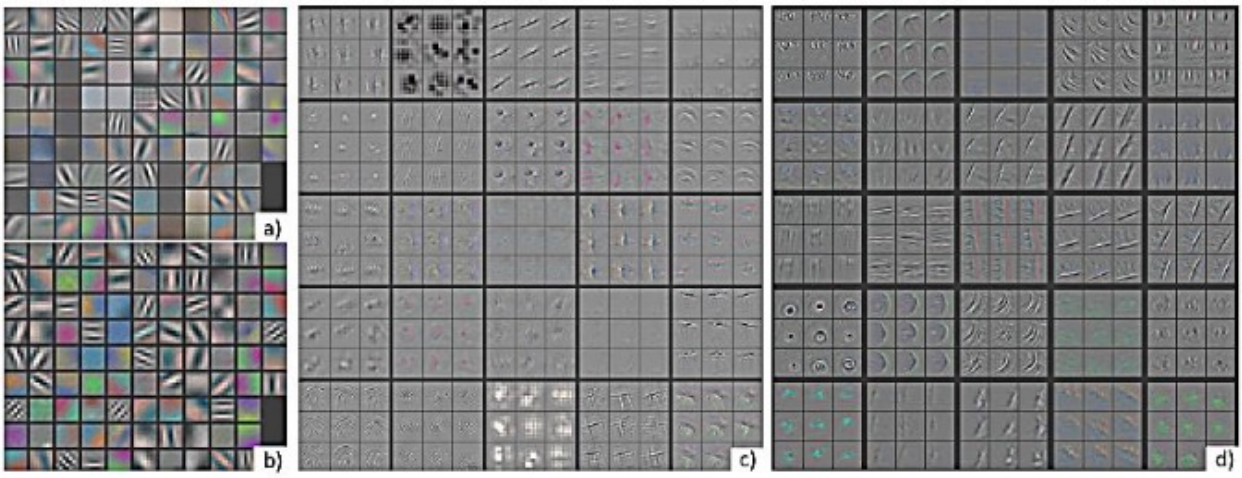


Fig 3. Convolutional kernel receptive field maps taken from a) AlexNet CONV1, b) ZFNet CONV1, c) AlexNet CONV2, ZFNet CONV2. CONV1 and CONV2 refer to first and second convolution modules, that received input from the original image. Already the original image is unrecognizable. (From Kaplanoglu, 2017)

It has been difficult to tease out precisely how a model is progressively trained to classify images, so as to provide step-by-step documentation as to the process of generating predictions. One can only point to performance indicators on other datasets, and infer that the accuracy and precision are replicated. This will likely be a strong reason for the argument that deep learning models will not replace a pathologist, but may play an assistive role to a trained pathologist, as long the pathologist does not second-guess his/her own reading, and place undue reliance on an erroneous reading of the trained model (Zhang, 2020).

Conclusions

Using a CNN designed for accuracy and efficiency, and designed to be compact enough to run on mobile devices, we were able to design and train a model on a dataset of breast histopathological images, such that it was able to predict the classification of the images as being benign vs malignant in nature, with a high degree of precision and accuracy. It was also moderately successful in classifying the pathology into one of eight subcategories.

This process did not require special feature extraction techniques, nor did it require prior semantic segmentation. The fine-tuning process was greatly aided by automating the training process, to heuristically identify the optimal hyperparameter settings. Overfitting was addressed with data augmentation as well as using Dropout and BatchNormalization. The model was trained on computers available on the consumer market.

Our achievement is a testament to the progress made in computer vision deep learning since the debut of AlexNet in 2012. As these models improve further, and as pathologists develop trust in their accuracy, trained CNNs could become a useful addition to the pathologist's diagnostic tookit.

References and Bibliography

American Cancer Society. (2023) Key Statistics for Breast Cancer. https://www.cancer.org/cancer/types/breast-cancer/about/how-common-is-breast-cancer.html (Accessed 13 August 2023)

Amerikanos, P., Maglogiannis, I. (2022). Image Analysis in Digital Pathology Utilizing Machine Learning and Deep Neural Networks. J Pers. Med. 12(9):1444. https://doi.org/10.3390/jpm12091444

Araújo, T., Aresta, G., Castro, E., Rouco, J., Aguiar, P., Eloy, C., Polónia, A., Campilho, A. (2017). Classification of breast cancer histology images using Convolutional Neural Networks. PLoS ONE 12(6): e0177544. https://doi.org/10.1371/journal.pone.0177544

Campanella, G., Hanna, M.G., Geneslaw, L., Miraflor, A., Silva, V.W.K., Busam, K.J., Brogi, E., Reuter, V.E., Klimstra, D.S., Fuchs1, T.J. (2019). Clinical-grade computational pathology using weakly supervised deep learning on whole slide image. Nat Med. 25(8):1301–1309. https://doi.org/10.1038/s41591-019-0508-1

Howard, A., Sandler, M., Chu, G., Chen, L-C., Chen B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H. (2019). Searching for MobileNetV3. Conference: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/ICCV.2019.00140

Janowcyzk, A., Madabhushi, A. (2016). Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases. J Pathol Informatics, 7(1):29. https://doi.org/10.4103/2153-3539.186902

Joshi, S.A., Bongale, A.M., Olsson P.O., Urolagin, S., Dharro, D., Bongale, A. (2023). Enhanced Pre-Trained Xception Model Transfer Learned for Breast Cancer Detection. Computation. 11(3):59. https://doi.org/10.3390/computation11030059

Kaplanoglu, P.I. (2017). Content-Based Image Retrieval using Deep Learning. Master's Thesis. Alexander Technological Educational Institute of Thessaloniki, Department Of Information Technology. https://doi.org/10.13140/RG.2.2.29510.16967

Krizhevsky, A., Sutskever, I., Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Li, C.I., Anderson, B.O., Daling J.R. (2003). Trends in Incidence Rates of Invasive Lobular and Ductal Breast Carcinoma. JAMA. 289(11):1421-1424. https://doi.org/10.1001/jama.289.11.1421

Pandey, A., (2018). Depth-wise convolution and depth-wise separable convolution. Medium. https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Russel, S., & Norvig, P. (2020). *Artificial Intelligence A Modern Approach* (4th ed., p. 121). Pearson.

Spanhol, F., Oliveira, L.S., Petitjean, C., Heutte, L. (2016). A Dataset for Breast Cancer Histopathological Image Classification, IEEE Transactions on Biomedical Engineering (TBME), 63(7):1455-1462. https://web.inf.ufpr.br/vri/databases/breast-cancer-histopathological-database-breakhis/

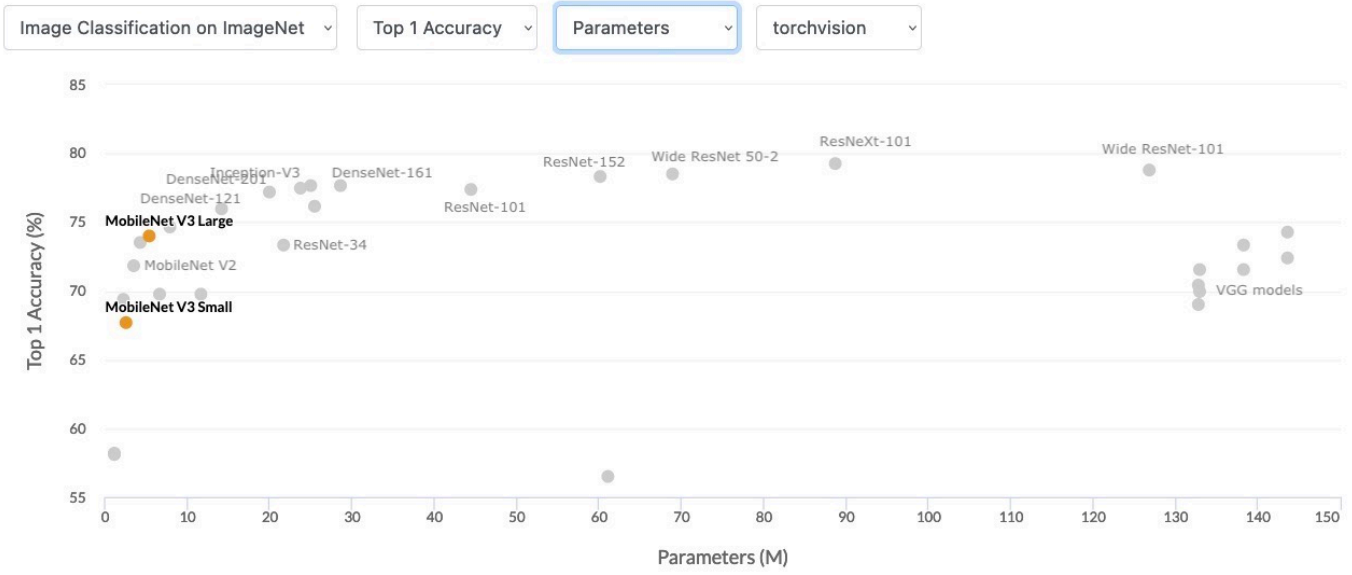
Weigelt, B., Geyer, F.C., Reis-Filho, J.S. (2010). Histological types of breast cancer: How special are they? Mol Oncol. 2010 Jun; 4(3): 192–208 https://doi.org/10.1016/j.molonc.2010.04.004

Zhang, Y., Liao, Q.V., Bellamy, R.K.E. (2020). Effect of Confidence and Explanation on Accuracy and Trust Calibration in AI-Assisted Decision Making. In Conference on Fairness, Accountability, and Transparency (FAT* '20), January 27–30, 2020, Barcelona, Spain. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3351095.3372852

Zhu C., Song, F., Wang, Y., Dong, H., Guo, Y., Liu, J. (2019). Breast cancer histopathology image classification through assembling multiple compact CNNs. BMC Med Inform Decis Mak 19: 198. https://doi.org/10.1186/s12911-019-0913-x.

Appendix 1 – Comparison of CNN architectures

	Year	Layers	Parameters
LeNet-5	1998	7	60,000
AlexNet	2012	8	62M
VGG16	2014	16	138M
ResNet-101	2015	152	44.7M
ResNet-152	2015	152	60.3M
DenseNet-121	2016	121	8.1M
InceptionV3	2016	311	6.4M
DenseNet-201	2016	201	29M
Xception-71	2017	171	37.3M
Inception-V4	2016	19	42.7M
MobileNet V1	2017	28	4.24M
MobileNet V2	2018	155	3.47
MobileNet V3 Small	2019	229	2.9M
MobileNet V3 Large	2019	263	5.4M



Adopted from MobileNet V3. https://paperswithcode.com/lib/torchvision/mobilenet-v3

Appendix 2

August 13, 2023

0.1 Setup

0.1.1 Library Imports

[1]:
```
import itertools
from itertools import cycle
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import interp

from sklearn.metrics import auc, confusion_matrix, f1_score, precision_score,␣
 ↪recall_score, roc_curve
from sklearn.preprocessing import LabelBinarizer

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import imagenet_utils
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image, image_dataset_from_directory

%matplotlib inline
```

0.1.2 Define Constants

[2]:
```
BATCH_SIZE = 32
CLASSES = ['adenosis', 'ductal_carcinoma', 'fibroadenoma', 'lobular_carcinoma',␣
 ↪'mucinous_carcinoma',
           'papillary_carcinoma', 'phyllodes_tumor', 'tubular_adenoma']
CLASSES_BINARY = ['benign', 'malignant']
IMG_SIZE = (224, 224)
IMG_SHAPE = IMG_SIZE + (3,)
PARAMETER_FILE = 'RUN_SET.csv' # File containing parameters to run - results␣
 ↪will be saved to same file
SEED = 42
```

```
TRAIN_SPLIT = 0.75
VAL_TEST_SPLIT = 0.6
```

0.1.3 Dataset Creation

[3]:
```
path_to_files = 'BreaKHis_v1/histology_slides/breast/'

train_dataset, holdout_dataset = tf.keras.preprocessing.
 ↪image_dataset_from_directory(
  path_to_files,
  labels = 'inferred',
  validation_split=(1-TRAIN_SPLIT),
  subset="both",
  seed=SEED,
  image_size=IMG_SIZE,
  batch_size=BATCH_SIZE,
  shuffle=True
)

holdout_batches = len(holdout_dataset)
val_size = int(holdout_batches*VAL_TEST_SPLIT)
test_size = int(holdout_batches*(1-VAL_TEST_SPLIT))

val_dataset = holdout_dataset.take(val_size)
test_dataset = holdout_dataset.skip(val_size).take(test_size)

print("Train dataset length: ", len(train_dataset))
print("Val dataset length: ", len(val_dataset))
print("Test dataset length: ", len(test_dataset))

# Use AUTOTUNE to decrease I/O roadblocks
train_dataset = train_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
val_dataset = val_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
Found 7909 files belonging to 8 classes.
Using 5932 files for training.
Using 1977 files for validation.
Train dataset length:  186
Val dataset length:  37
Test dataset length:  24
```

0.2 Model Definition and Training

0.2.1 Model Definition

[4]:
```
preprocess_input = tf.keras.applications.mobilenet_v3.preprocess_input
mobile_v3 = tf.keras.applications.MobileNetV3Large()  # load the model with␣
 ↪pretrained weights

def data_augmenter():
    return tf.keras.Sequential([
        tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
    ])

def new_model (image_shape=IMG_SIZE, fine_tune_at=150):
    input_shape = image_shape + (3,)
    base_model = tf.keras.applications.MobileNetV3Large(input_shape=IMG_SHAPE,
                                                        include_top=False, #␣
 ↪important
                                                        weights='imagenet')

    # freeze the base model
    base_model.trainable = True

    # freeze all the layers before the `fine_tune_at` layer
    for layer in base_model.layers[:fine_tune_at]:
        layer.trainable = False

    # create input layer
    inputs = tf.keras.Input(shape=input_shape)
    # pre-process inputs
    x = preprocess_input(inputs)

    x = data_augmenter()(x)
    # set training to False to avoid tracking statistics in batch norm layer
    x = base_model(x, training=False)

    # add custom layers
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    x = tf.keras.layers.Dense(64, activation='relu')(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dense(32, activation='relu')(x)
    x = tf.keras.layers.Dense(16, activation='relu')(x)
    x = tf.keras.layers.Dense(8, activation='softmax')(x)

    outputs = x

    model = tf.keras.Model(inputs,outputs)
```

```
    return model
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not
224. Weights for input shape (224, 224) will be loaded as the default.
```

0.2.2 Model Training

[5]:
```
def train_model(model, train_dataset, val_dataset, params):

    # Unpack parameters
    num_epochs = params['num_epochs']
    lr = params['lr']
    epoch_decay = params['epoch_decay']

    steps_per_epoch = len(train_dataset)

    lr_schedule = tf.keras.optimizers.schedules.
↪ExponentialDecay(initial_learning_rate=lr, decay_steps=steps_per_epoch,␣
↪decay_rate=epoch_decay)

    optimizer = tf.keras.optimizers.legacy.SGD(learning_rate=lr_schedule,␣
↪momentum=0.95, nesterov=True)

    model.compile(optimizer=optimizer,
                  loss=tf.keras.losses.
↪SparseCategoricalCrossentropy(from_logits=False),
                  metrics=['accuracy'])

    history = model.fit(
        train_dataset,
        epochs=num_epochs,
        verbose=1,
        validation_data=val_dataset)

    return history
```

0.2.3 Training Loop

Loop through the hyperparameters in the PARAMETER_FILE and train a model for each set of hyperparameters.

[6]:
```
# Training loop

run_set = pd.read_csv(PARAMETER_FILE)

for index, row in run_set.iterrows():
    if pd.isna(row['T1']):
```

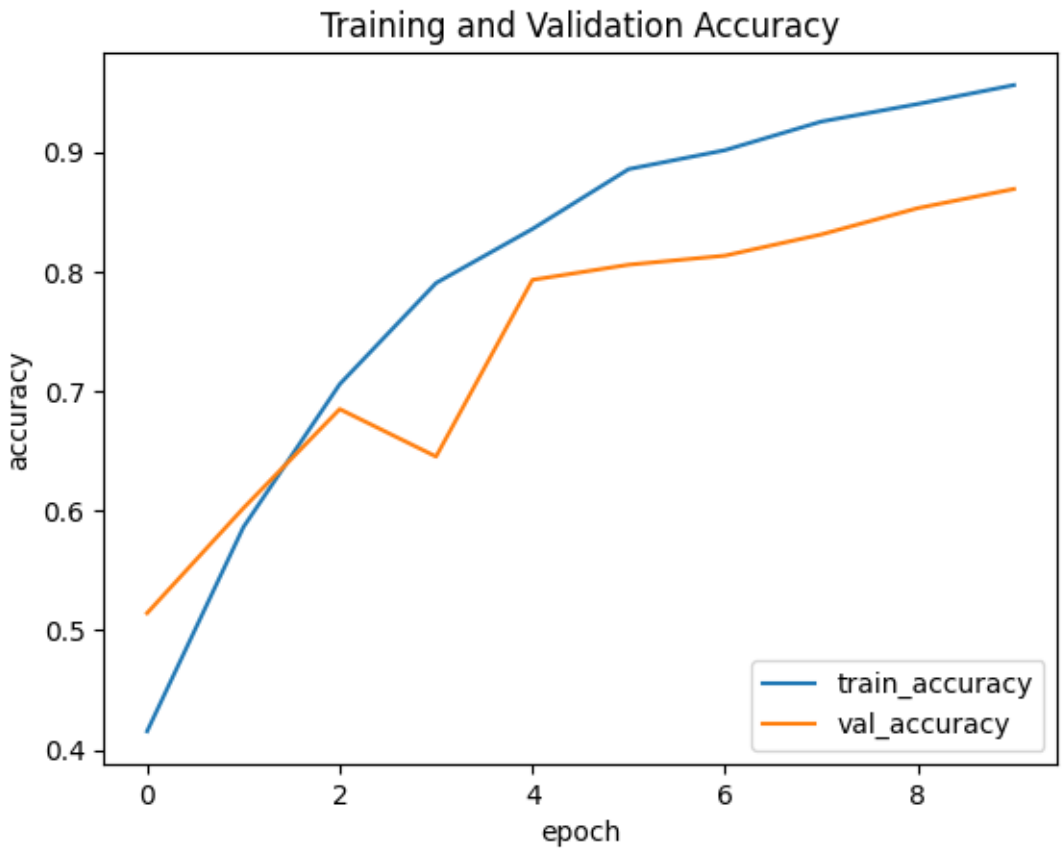
```
        print(f"Training model {index+1} of {len(run_set)}")
        print('-'*30)

        # Pack parameters
        params = {}
        params['num_epochs'] = int(row['num_epochs'])
        params['lr'] = row['lr']
        params['epoch_decay'] = row['epoch_decay']
        fine_tune_at = int(row['fine_tune_at'])

        # Create the model
        model = new_model(fine_tune_at=fine_tune_at)

        # Train the model
        history = train_model(model, train_dataset, val_dataset, params)

        # Write history to run set
        for epoch in range(params['num_epochs']):
            train_number = 'T' + str(epoch+1)
            run_set.at[index, train_number] = history.history['accuracy'][epoch]
            val_number = 'V' + str(epoch+1)
            run_set.at[index, val_number] = history.
↪history['val_accuracy'][epoch]

        # Save the run set
        run_set.to_csv(PARAMETER_FILE, index=False)

        print('-'*30)
        print(f"Model {index+1} of {len(run_set)} trained and saved")
        print('-'*30)
        print()
```

```
Training model 75 of 75
------------------------------
Epoch 1/10
186/186 [==============================] - 82s 430ms/step - loss: 1.6523 -
accuracy: 0.4155 - val_loss: 1.4710 - val_accuracy: 0.5144
Epoch 2/10
186/186 [==============================] - 80s 427ms/step - loss: 1.1917 -
accuracy: 0.5863 - val_loss: 1.0337 - val_accuracy: 0.6022
Epoch 3/10
186/186 [==============================] - 79s 421ms/step - loss: 0.8503 -
accuracy: 0.7058 - val_loss: 0.8496 - val_accuracy: 0.6850
Epoch 4/10
186/186 [==============================] - 79s 424ms/step - loss: 0.6238 -
accuracy: 0.7905 - val_loss: 1.0605 - val_accuracy: 0.6453
Epoch 5/10
```

```
186/186 [==============================] - 80s 426ms/step - loss: 0.4675 -
accuracy: 0.8356 - val_loss: 0.6002 - val_accuracy: 0.7931
Epoch 6/10
186/186 [==============================] - 79s 422ms/step - loss: 0.3342 -
accuracy: 0.8857 - val_loss: 0.5241 - val_accuracy: 0.8057
Epoch 7/10
186/186 [==============================] - 78s 420ms/step - loss: 0.2737 -
accuracy: 0.9016 - val_loss: 0.5038 - val_accuracy: 0.8133
Epoch 8/10
186/186 [==============================] - 79s 421ms/step - loss: 0.2238 -
accuracy: 0.9255 - val_loss: 0.4626 - val_accuracy: 0.8311
Epoch 9/10
186/186 [==============================] - 79s 422ms/step - loss: 0.1715 -
accuracy: 0.9402 - val_loss: 0.4083 - val_accuracy: 0.8530
Epoch 10/10
186/186 [==============================] - 80s 427ms/step - loss: 0.1265 -
accuracy: 0.9560 - val_loss: 0.3588 - val_accuracy: 0.8691
------------------------------
Model 75 of 75 trained and saved
------------------------------
```

0.2.4 Training Accuracy

[7]:
```python
def plot_accuracy(train_accuracy, val_accuracy):
    # Plot the training and validation accuracy per epoch

    plt.plot(train_accuracy, label='train_accuracy')
    plt.plot(val_accuracy, label='val_accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.show()

plot_accuracy(history.history['accuracy'], history.history['val_accuracy'])
```



0.3 Evaluation

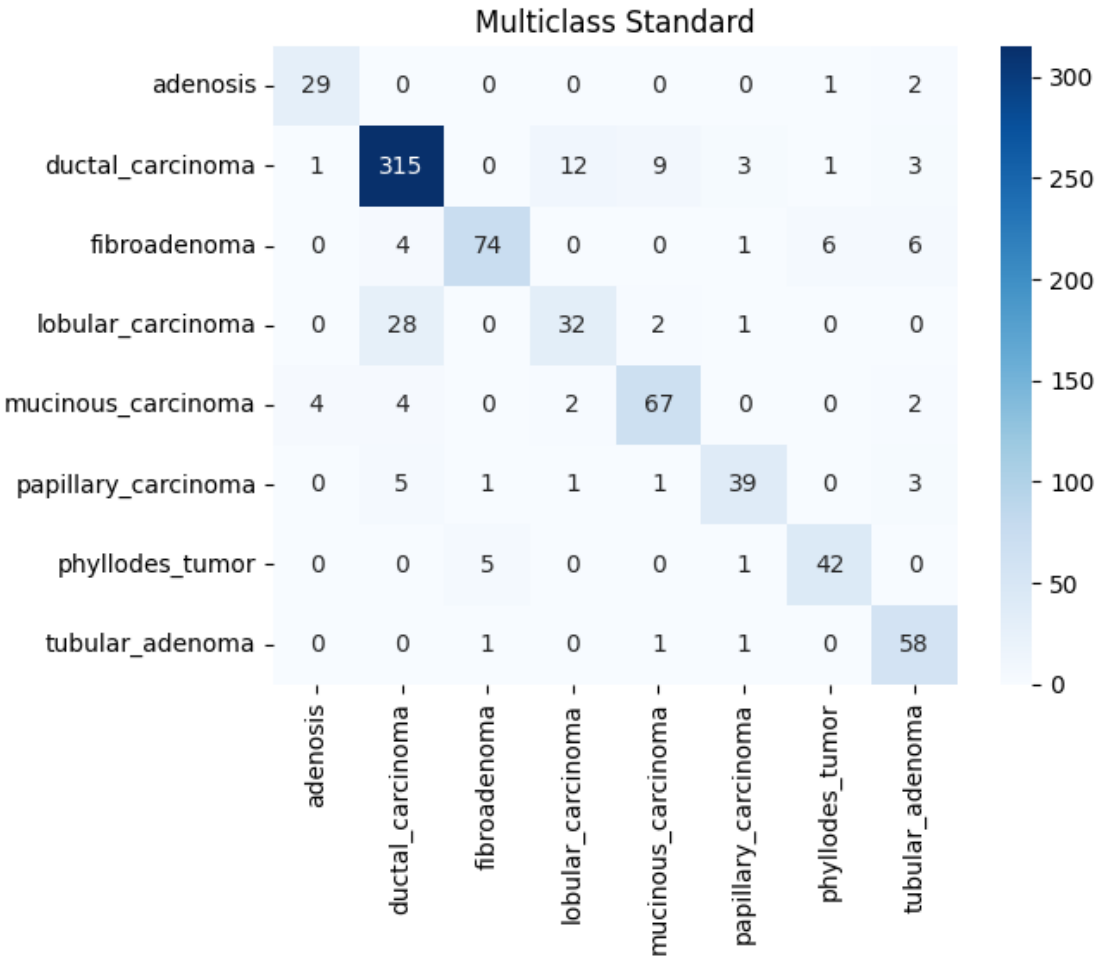
0.3.1 Collect Ground Truth and Generate Predictions

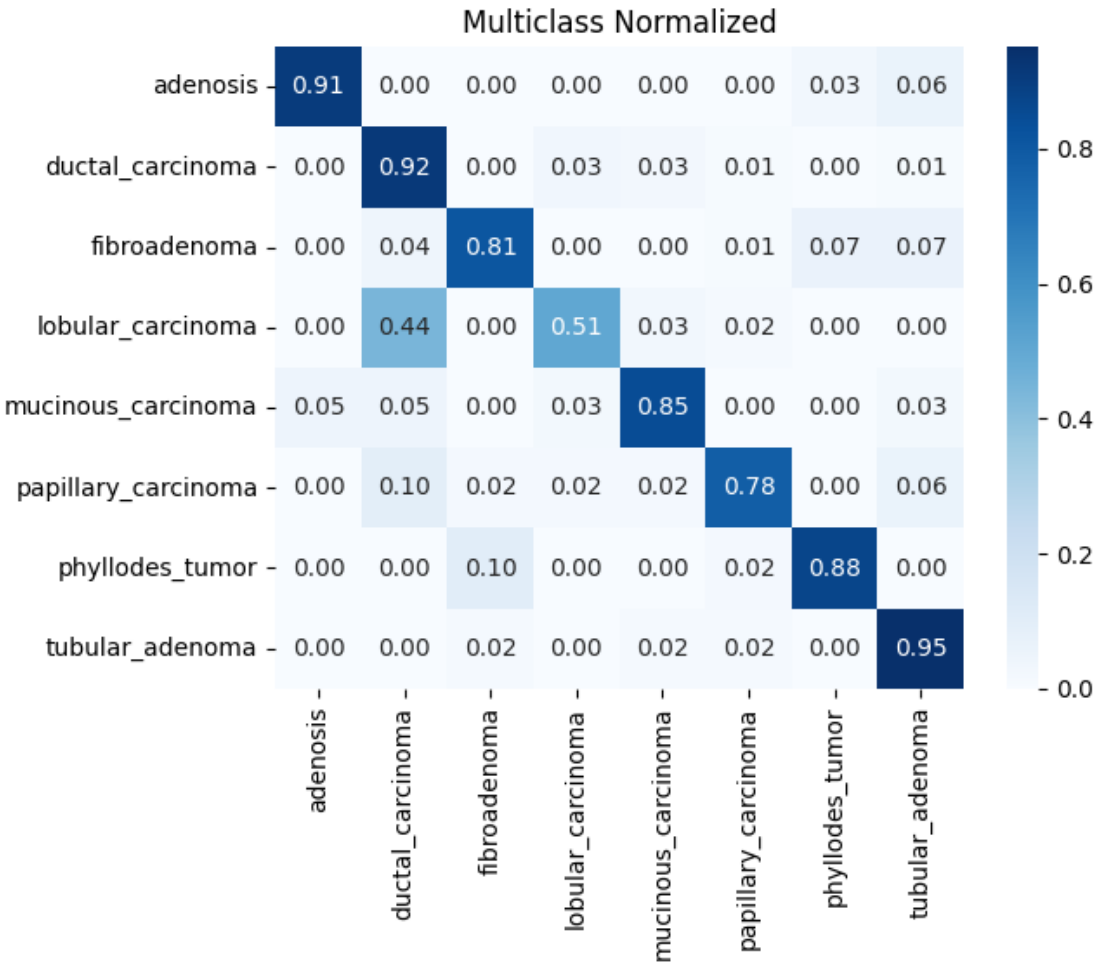
[8]:
```
# Gather the ground truth from the train set
ytrain = []
for images, labels in train_dataset.map(lambda x, y: (x, y)):
    ytrain += labels.numpy().tolist()
ytrain_count = [ytrain.count(i) for i in range(len(CLASSES))]

# Get ground truth from test set
ytest = []
for images, labels in test_dataset.map(lambda x, y: (x, y)):
    ytest += labels.numpy().tolist()
y_binary_test = [i in [1, 3, 4, 5] for i in ytest]

# Get predictions from test set
ypred_raw = model.predict(test_dataset)
ypred = ypred_raw.argmax(axis=-1).tolist()
y_binary_pred = [i in [1, 3, 4, 5] for i in ypred]
```

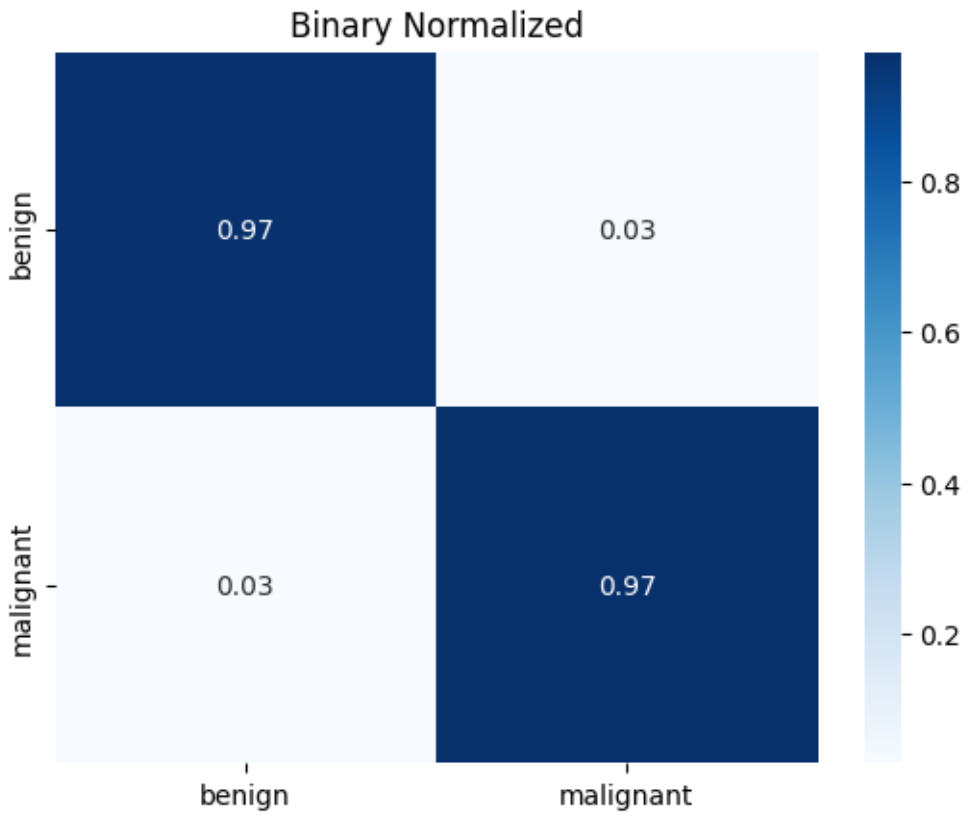
```
24/24 [==============================] - 7s 207ms/step
```

0.3.2 Confusion Matrices

[9]:
```
def generate_confusion_matrix(y_true, y_pred, classes, title, normalize=False):
    cm = confusion_matrix(y_true, y_pred)
    format = 'g'
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        format = '.2f'
    sns.heatmap(cm, annot=True, fmt=format, cmap='Blues', xticklabels=classes,␣
 ↪yticklabels=classes)
    plt.title(title)
    plt.show()

generate_confusion_matrix(ytest, ypred, CLASSES, 'Multiclass Standard',␣
 ↪normalize=False)
generate_confusion_matrix(ytest, ypred, CLASSES, 'Multiclass Normalized',␣
 ↪normalize=True)
generate_confusion_matrix(y_binary_test, y_binary_pred, CLASSES_BINARY, 'Binary␣
 ↪Normalized', normalize=True)
```


Multiclass Standard
adenosis
ductal_carcinoma
fibroadenoma
lobular_carcinoma
mucinous_carcinoma
papillary_carcinoma
phyllodes_tumor
tubular_adenoma
29 0 0 0 0 0 1 2
1 315 0 12 9 3 1 3
0 4 74 0 0 1 6 6
0 28 0 32 2 1 0 0
4 4 0 2 67 0 0 2
0 5 1 1 1 39 0 3
0 0 5 0 0 1 42 0
0 0 1 0 1 1 0 58
300
250
200
150
100
50
0


Multiclass Normalized
adenosis
ductal_carcinoma
fibroadenoma
lobular_carcinoma
mucinous_carcinoma
papillary_carcinoma
phyllodes_tumor
tubular_adenoma
0.91 0.00 0.00 0.00 0.00 0.00 0.03 0.06
0.00 0.92 0.00 0.03 0.03 0.01 0.00 0.01
0.00 0.04 0.81 0.00 0.00 0.01 0.07 0.07
0.00 0.44 0.00 0.51 0.03 0.02 0.00 0.00
0.05 0.05 0.00 0.03 0.85 0.00 0.00 0.03
0.00 0.10 0.02 0.02 0.02 0.78 0.00 0.06
0.00 0.00 0.10 0.00 0.00 0.02 0.88 0.00
0.00 0.00 0.02 0.00 0.02 0.02 0.00 0.95
0.8
0.6
0.4
0.2
0.0



0.3.3 F1, Precision and Recall Scores

[10]:
```
f1_score_all = f1_score(ytest, ypred, average=None)
precision_score_all = precision_score(ytest, ypred, average=None)
recall_score_all = recall_score(ytest, ypred, average=None)

# Compile the scores into a dataframe
scores_df = pd.DataFrame({'F1 score': f1_score_all, 'Precision score':␣
 ↪precision_score_all, 'Recall score': recall_score_all, 'Count':␣
 ↪ytrain_count}, index=CLASSES)
print(scores_df)

print("\nBenign vs Malignant Only:")
f1_score_binary = f1_score(y_binary_test, y_binary_pred)
print(f'F1: {f1_score_binary:.2f}')
precision_score_binary = precision_score(y_binary_test, y_binary_pred)
print(f'Precision: {precision_score_binary:.2f}')
recall_score_binary = recall_score(y_binary_test, y_binary_pred)
print(f'Recall: {recall_score_binary:.2f}')
```

```
                F1 score  Precision score  Recall score  Count
```

```
adenosis              0.878788        0.852941      0.906250    345
ductal_carcinoma      0.900000        0.884831      0.915698   2576
fibroadenoma          0.860465        0.913580      0.813187    753
lobular_carcinoma     0.581818        0.680851      0.507937    468
mucinous_carcinoma    0.842767        0.837500      0.848101    594
papillary_carcinoma   0.812500        0.847826      0.780000    437
phyllodes_tumor       0.857143        0.840000      0.875000    331
tubular_adenoma       0.859259        0.783784      0.950820    428

Benign vs Malignant Only:
F1: 0.98
Precision: 0.98
Recall: 0.97
```

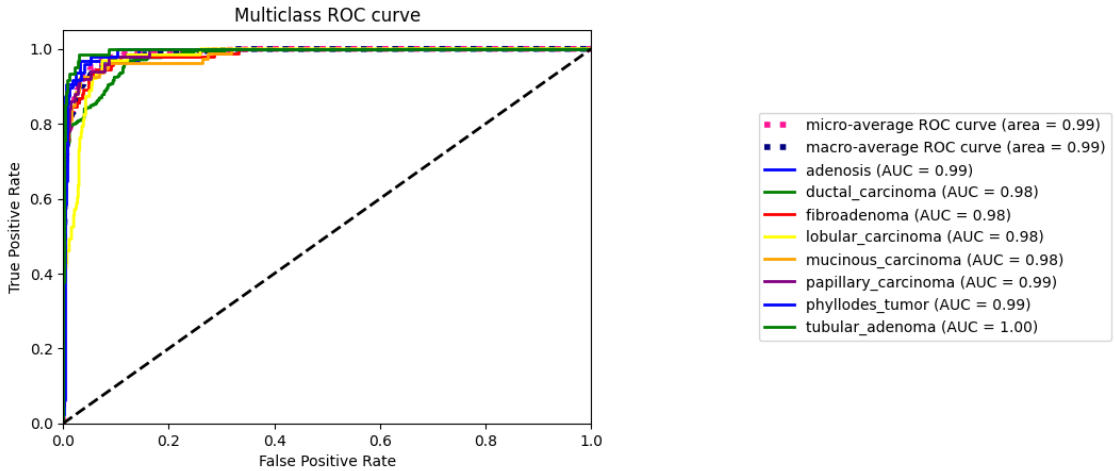
0.3.4 ROC Curves

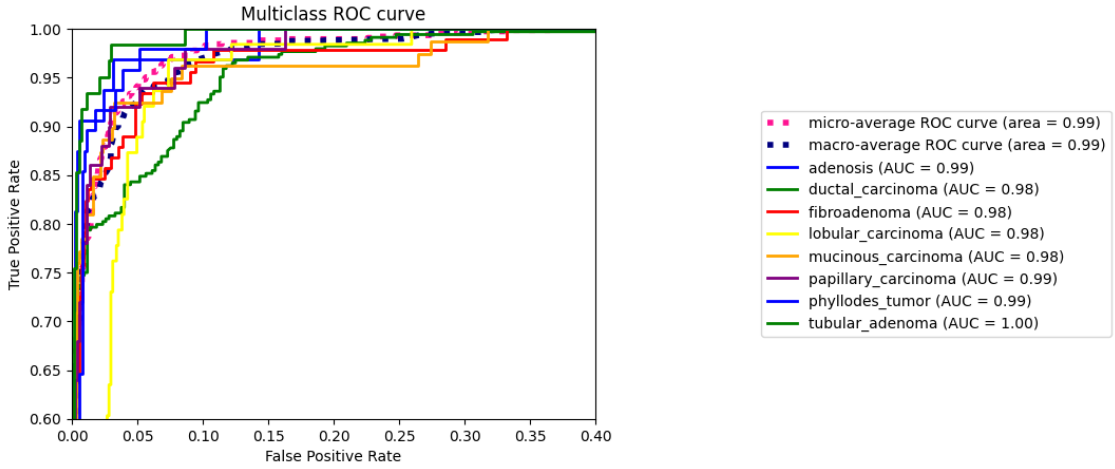
The following cells for generating the ROC curves were adapted from the following source: Zhang, C. (2018, April 21). Simple guide on how to generate ROC plot for Keras classifier. Medium; HackerNoon.com. https://medium.com/hackernoon/simple-guide-on-how-to-generate-roc-plot-for-keras-classifier-2ecc6c73115a

[11]:
```
lb = LabelBinarizer()
lb.fit(ytest)
ytest_lb = lb.transform(ytest)

# Compute ROC curve and ROC area for each class
fpr = {}
tpr = {}
roc_auc = {}
for i in range(len(CLASSES)):
    fpr[i], tpr[i], _ = roc_curve(ytest_lb[:, i], ypred_raw[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(ytest_lb.ravel(), ypred_raw.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# Compute macro-average ROC curve and ROC area

# First aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(len(CLASSES))]))

# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(len(CLASSES)):
    mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
```

```
mean_tpr /= len(CLASSES)

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
```

[12]:
```
def plot_roc(fpr, tpr, roc_auc, classes, zoom=False):

      # Generate micro-average ROC curve plot
      plt.plot(fpr["micro"], tpr["micro"],
            label='micro-average ROC curve (area = {0:0.2f})'
                  ''.format(roc_auc["micro"]),
            color='deeppink', linestyle=':', linewidth=4)

      # Generate macro-average ROC curve plot
      plt.plot(fpr["macro"], tpr["macro"],
            label='macro-average ROC curve (area = {0:0.2f})'
                  ''.format(roc_auc["macro"]),
            color='navy', linestyle=':', linewidth=4)

      # Generate ROC curve for each class
      colors = cycle(['blue', 'green', 'red', 'yellow', 'orange', 'purple'])
      for i, color in zip(range(len(CLASSES)), colors):
            plt.plot(fpr[i], tpr[i], color=color, lw=2,
                  label=f'{CLASSES[i]} (AUC = {roc_auc[i]:0.2f})')

      # Generate line for random guess
      plt.plot([0, 1], [0, 1], 'k--', lw=2)

      # Format and display plot
      if zoom:
            plt.xlim(0.0, 0.4)
            plt.ylim(0.6, 1.0)
      else:
            plt.xlim(0.0, 1.0)
            plt.ylim(0.0, 1.05)
      plt.xlabel('False Positive Rate')
      plt.ylabel('True Positive Rate')
      plt.title('Multiclass ROC curve')
      if zoom:
            plt.legend(bbox_to_anchor=(2.0, 0.5), loc='center right')
      else:
            plt.legend(bbox_to_anchor=(2.0, 0.5), loc="center right")
      plt.show()

plot_roc(fpr, tpr, roc_auc, CLASSES, zoom=False)
plot_roc(fpr, tpr, roc_auc, CLASSES, zoom=True)
```





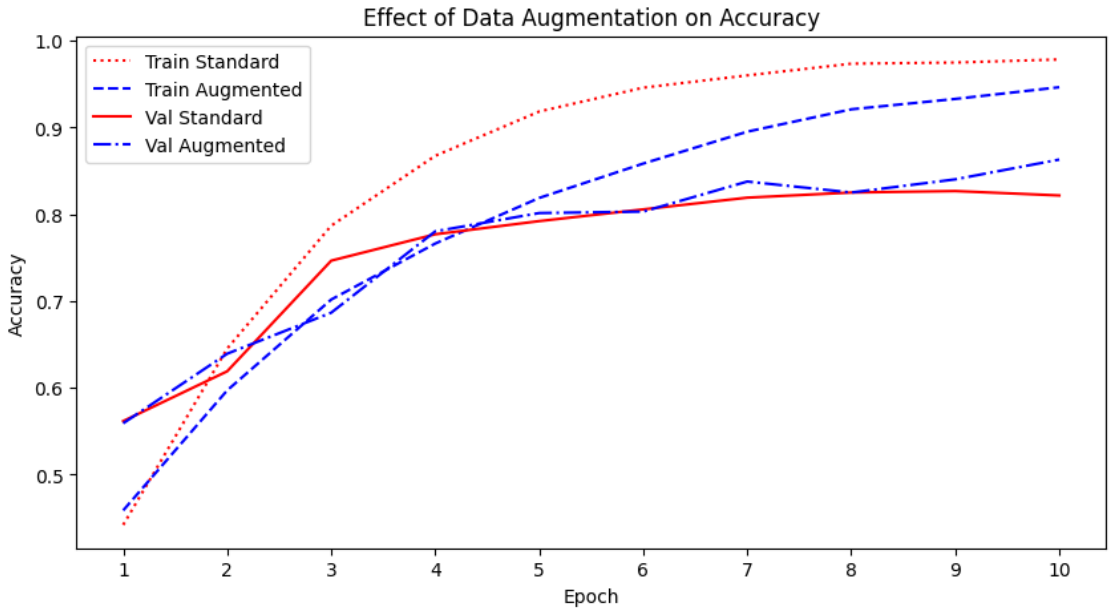
0.4 Additional Tuning

0.4.1 Data Augmentation

Study that shows the effect of data augmentation. Simple horizontal flips of the images during training reduced overfitting and improved validation accuracy.

[13]:
```
standard_row = 28
augmented_row = 29

train_standard = []
train_augmented = []
val_standard = []
val_augmented = []
```

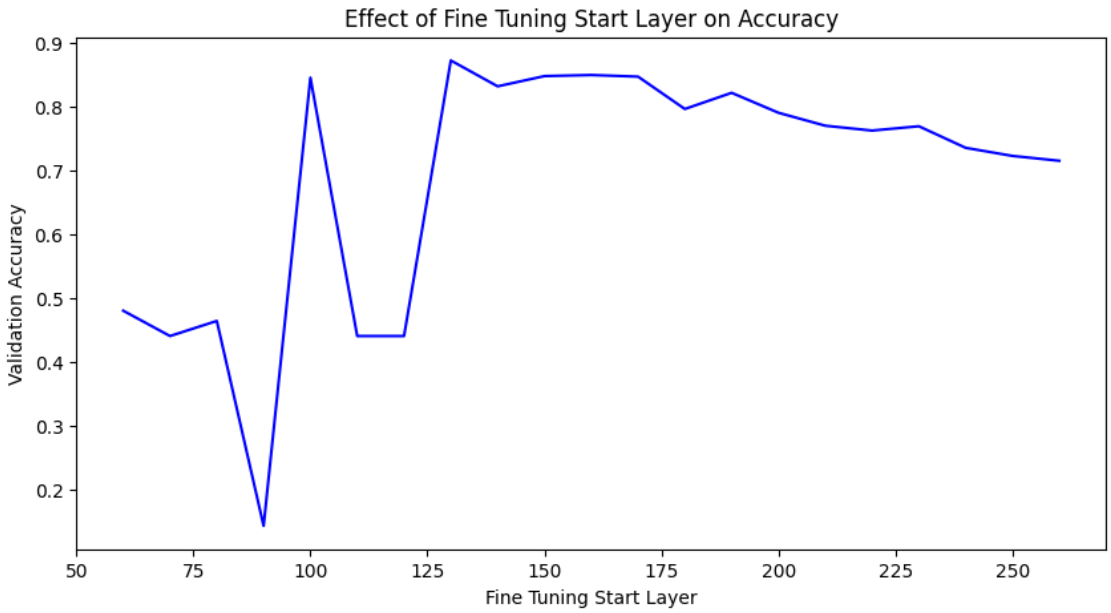
```
# Collect data from the standard and augmented runs
for i in range(10):
    train_label = 'T' + str(i+1)
    val_label = 'V' + str(i+1)
    train_standard.append(run_set.iloc[standard_row][train_label])
    train_augmented.append(run_set.iloc[augmented_row][train_label])
    val_standard.append(run_set.iloc[standard_row][val_label])
    val_augmented.append(run_set.iloc[augmented_row][val_label])

# Plot the results
plt.figure(figsize=(10, 5))
x = np.arange(1, 11)
plt.plot(x, train_standard, label='Train Standard', color='red', linestyle=':')
plt.plot(x, train_augmented, label='Train Augmented', color='blue',␣
 ↪linestyle='--')
plt.plot(x, val_standard, label='Val Standard', color='red', linestyle='-')
plt.plot(x, val_augmented, label='Val Augmented', color='blue', linestyle='-.')
plt.xticks(x)
plt.title('Effect of Data Augmentation on Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



0.4.2 Layer Freezing

Study to determine the optimal layers to freeze for transfer learning. Fine tuning start layer represents the first layer that is unfrozen.

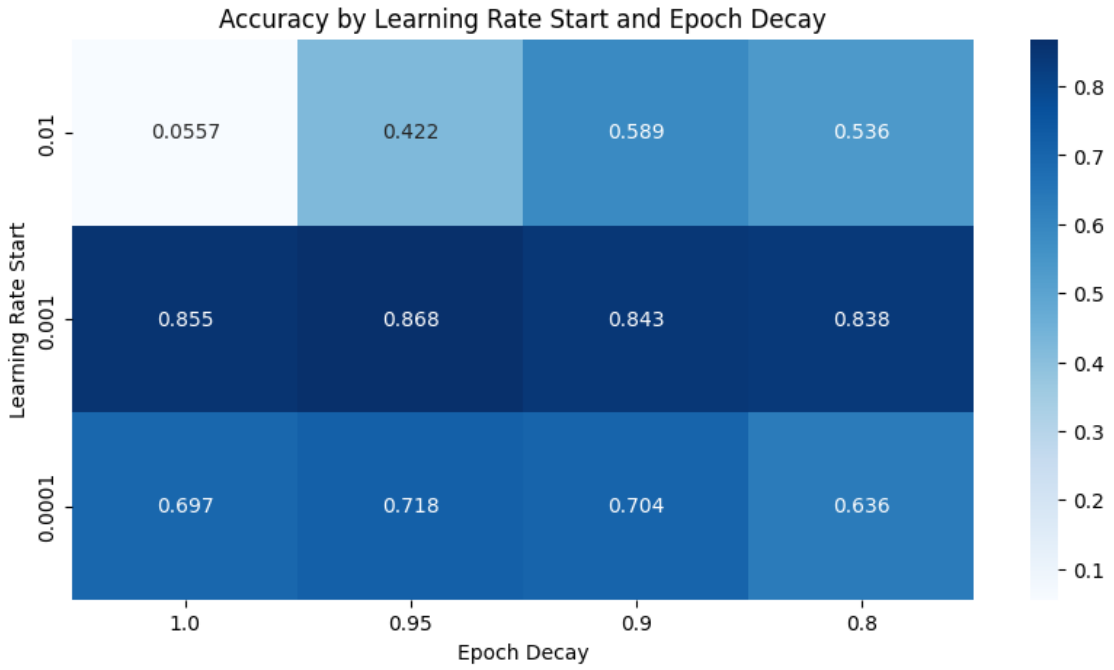
[14]:
```
freeze_study_start = 40
freeze_study_end = 61

# Collect data from the fine tuning layer freeze runs
accuracy_scores = run_set.iloc[freeze_study_start:freeze_study_end]['V10'].
 ↪tolist()
layer_starts = run_set.iloc[freeze_study_start:
 ↪freeze_study_end]['fine_tune_at'].tolist()
accuracy_scores.reverse()
layer_starts.reverse()

# Plot the results
plt.figure(figsize=(10, 5))
plt.plot(layer_starts, accuracy_scores, color='blue')
plt.title('Effect of Fine Tuning Start Layer on Accuracy')
plt.xlabel('Fine Tuning Start Layer')
plt.ylabel('Validation Accuracy')
plt.show()
```



0.4.3 Learning Rate Selection

Study to determine the best initial learning rate and decay rate for the model.

[15]:
```
lr_study_start = 61
lr_study_end = 73

# Collect the results from the run set
lr_results_list = run_set.iloc[lr_study_start:lr_study_end]['V10'].tolist()
lr_results = np.reshape(lr_results_list, (3, 4))
epoch_decay_list = run_set.iloc[lr_study_start:lr_study_end]['epoch_decay'].
 ↪tolist()
lr_start_list = run_set.iloc[lr_study_start:lr_study_end]['lr'].tolist()

# Get unique values from the epoch decay and lr_start lists
epoch_decay = sorted(list(set(epoch_decay_list)), reverse=True)
lr_start = sorted(list(set(lr_start_list)), reverse=True)

# Display the results in a heatmap
fig, ax = plt.subplots(figsize=(10, 5))
sns.heatmap(lr_results, annot=True, xticklabels=epoch_decay,␣
 ↪yticklabels=lr_start, fmt='.3g', cmap='Blues', ax=ax)
ax.set_xlabel('Epoch Decay')
ax.set_ylabel('Learning Rate Start')
ax.set_title('Accuracy by Learning Rate Start and Epoch Decay')
plt.show()
```



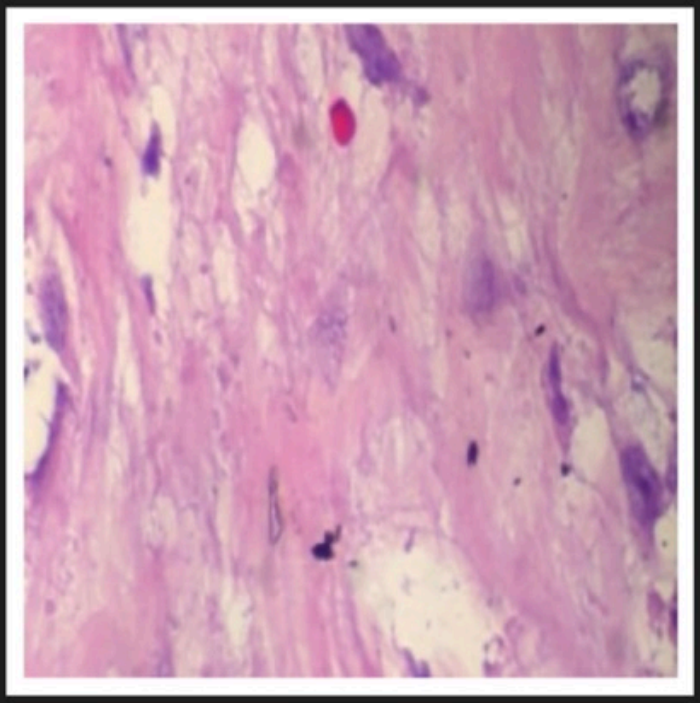
0.5 Misclassified Images

Samples first n number of requested misclassified images from the test set.

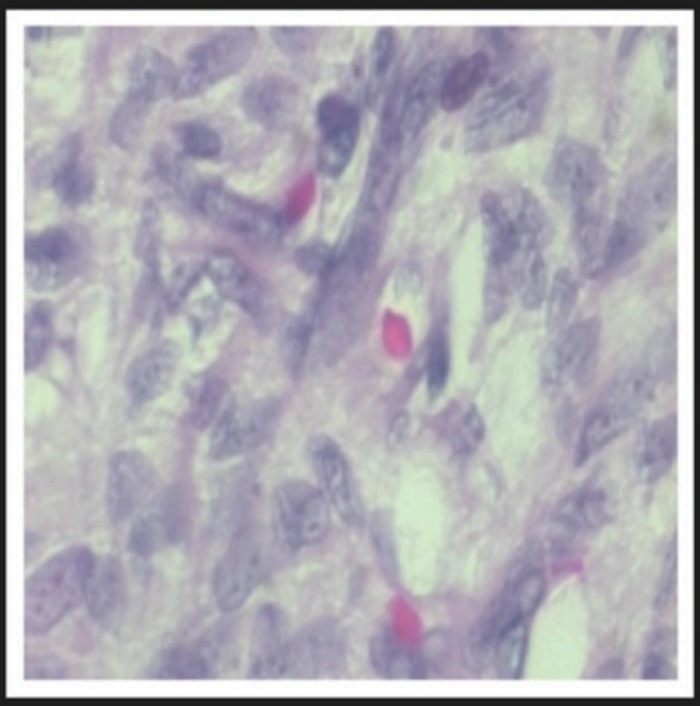
[16]:
```
def display_misclassified(classes, ypred, ytest, test_dataset, requested=5):
    found = i = 0
    while found < requested and i < len(ypred):
        if ypred[i] != ytest[i]:
            found += 1

            batch_num = i // BATCH_SIZE
            image_num = i % BATCH_SIZE

            batch = test_dataset.skip(batch_num).take(1)
            for images, labels in batch:
                images = images
                labels = labels

            image = images[image_num].numpy()

            print("Actual: ", classes[labels[image_num]])
            print("Predicted: ", classes[ypred[i]])

            # Print the image from numpy array
            plt.imshow(image.astype("uint8"))
            plt.axis("off")
            plt.show()

        i += 1

display_misclassified(CLASSES, ypred, ytest, test_dataset)
```

```
Actual:  ductal_carcinoma
Predicted:  lobular_carcinoma
```

Actual: ductal_carcinoma
Predicted: lobular_carcinoma



Actual: fibroadenoma
Predicted: phyllodes_tumor



Actual: fibroadenoma
Predicted: phyllodes_tumor

Actual: lobular_carcinoma
Predicted: ductal_carcinoma