A Novel Application of Different-Sized Deep Residual Networks (ResNet) to Breast Cancer Identification

Abstract

Invasive ductal carcinoma (IDC) is the most common subtype of breast cancer, and it is important to identify it as quickly and accurately as possible. Currently, identification of IDC is done by hand, which is time-consuming and challenging. Machine learning, specifically convolutional neural networks (CNNs) that specialize in computer vision, can be applied to assist in the diagnosis process. ResNet (residual neural network) is a type of pre-trained CNN that utilizes transfer learning. When a model implements ResNet, it is able to train on a smaller dataset, which is highly relevant to training models with applications in healthcare due to limited access to data. Currently, there is a gap in knowledge regarding the effect of a ResNet's depth on the model's performance in cancer identification. This paper hypothesized that ResNet18 would perform the worst in identifying IDC, while ResNet101 would perform the best and tested the performance of four models, each with ResNets of different depths implemented in them. In this project, the dataset, a collection of slide images with IDC, was obtained from Kaggle, downsampled to address class imbalance, and pre-processed. Then, four models using different-sized ResNets (ResNet18, ResNet34, ResNet50, and ResNet101) were trained and evaluated. Across all sets (training, validation, and testing), ResNet18 performed the worst (F1-score of 0.5816 in testing), while ResNet101 performed the best (F1-score of 0.7249 in testing). As the depth of the ResNet increased, the higher it scored in performance, measured by an F1-score. This could be explained due to the fact that, generally, the deeper the neural network is, the more powerful it is and the better it performs. Another result was that the validation accuracy for all models was unstable. This could be the result of insufficient training and is an improvement that could be made in future research. An interesting point of future research could be comparing the ability of different types of pre-trained networks other than ResNet in identifying cancer. With further training and hyperparameter optimization of the models, there is still great potential for real world application, especially in assisting in the cancer identification process. It is crucial to study pre-trained models due to the fact that they reduce the amount of data needed to train a model, and data is difficult to access in healthcare.

Introduction

According to the World Health Organization, in 2020, female breast cancer was the most commonly diagnosed cancer, with an estimated 2.3 million new cases (Sung et al., 2021). Of the phenotypic subtypes of breast cancers, invasive ductal carcinoma (IDC) is the most common, accounting for 80 percent of subtypes (Cruz-Roa et al., 2014). IDC begins in the milk duct of the breast and spreads to surrounding fibrous and fatty tissue, and early diagnosis improves the chances of long-term survival (Page, 2003). Therefore, it is important to identify IDC as quickly and accurately as possible so that the aggressiveness of the cancer can be graded. The first step in grading the aggressiveness of a whole-mount sample is to identify the exact regions of each slide image that contain IDC. Currently, identification of IDC is performed by pathologists manually scanning large areas of benign regions to identify the areas of malignancy, an often time-consuming and challenging process (Cruz-Roa et al., 2014).

In recent years, there has been a growing interest in applying machine learning to various fields of healthcare, such as cancer diagnosis. This brings many advantages including a faster diagnosis process and checking for human-made errors. Doctor-assisting models have also become a topic of interest in machine learning. Deep learning, a field of machine learning that focuses on neural networks, has gained attention due to its success in different computer vision and pattern recognition tasks. Deep learning architectures have shown to be successful in the automated segmentation and classification of diseases (Cruz-Roa et al., 2014). A specific type of neural network, convolutional neural networks (CNNs) are well-suited for classifying images as they are very strong in pattern recognition on two-dimensional data, such as images and videos (Albawi et al., 2017). CNNs can be applied to slide images of breast cancer specimens to identify IDC.

A notable strength of CNNs is that they are translation invariant. Opposed to multilayer perceptrons, CNNs view the pixels of an image in relation to nearby pixels. This means that it can look for features in different locations. Using a filter, the CNN looks for a specific feature, like a snout in pictures of dogs, and notes the number of times and in what locations the feature appears in (Albawi et al., 2017).

The following 4 layers are common among CNNs: convolutional, non-linearity, pooling, and classification. In the first operation, the convolution operator extracts features from the input image by preserving the relationship between pixels. This allows it to learn image features. The purpose of the second layer, non-linearity, is to introduce non-linearity to the CNN because most real-world data is non-linear. This is done by applying ReLU, which replaces all negative pixel values in the feature map with zero. The pooling, or downsampling, layer reduces the dimensions of each feature map but retains the most important information. The purpose of the previous layers was to create outputs that represent the features of the input image. The classification, or fully connected layer, uses these outputs to classify the input image into classes. For each class, this layer outputs the probability that the input image belongs to it, and the final output is the class that the input image most likely belongs to.

When building models, transfer learning is often utilized. Transfer learning uses pre-trained models, which are models created by another researcher to solve a similar problem. The weights, which represent what the pre-trained model has learned, are used in the user's model, and the user's model can then train on a more specific dataset using limited data. Pre-trained models are used commonly as they reduce the amount of training data needed and the amount of time it takes for the model to learn. One well-known pre-trained CNN model is ResNet, which uses weights based on the ImageNet dataset. ResNet uses skip connection to fit the input from the previous layer to the next layer without any modifications. This allows it to have deeper networks (He et al., 2016). There are ResNets of different sizes with different numbers of layers. For example, ResNet50, one of the most popular ResNets, has 50 layers. ResNets come in the following sizes: 18, 34, 50, 101, and 152. Generally, the deeper the ResNet is (the more layers it has), the more powerful it is in its image recognition and classification abilities (He et al., 2016).

The use of CNNs in classifying images of cancer is common. However, there are currently no studies analyzing the performance of pre-trained networks of different sizes, specifically ResNet, in classifying cancer images. For example, Alazani et al. assessed the application of CNNs in mammograms to assist in diagnosing breast cancer. They explored using CNNs of different depths, and the deepest model performed the best (Alanazi et al., 2021). However, this study used CNNs developed by the authors and did not address pre-trained models. Similar studies can be found, but none of them address using pre-trained models of different sizes, specifically ResNet, in cancer classification. It is important to analyze the impact that model size has on performance, especially classification accuracy, as it is crucial to diagnose patients accurately and efficiently, and it is imperative to analyze the performance of pre-trained models in a healthcare setting because pre-trained models reduce the amount of data needed and data is difficult to access in healthcare due to patient privacy concerns.

This paper addresses this gap in knowledge by aiming to compare the performance of ResNets of different sizes (18-layer, 34-layer, 50-layer, and 101-layer) by using whole-mount slide images of breast cancer specimens that do or do not have invasive ductal carcinoma. It is hypothesized that the deepest model (ResNet101) will be more accurate in identifying IDC than the smaller models (ResNet18, ResNet34, and ResNet50).

Methods and Materials

The data used in this project were obtained from a publicly available dataset on Kaggle called "Breast Histopathology Images". The dataset contained 162 whole mount slide images of breast cancer specimens scanned at 40x. From that, 277,524 patches of size 50 x 50 were extracted. In total, 198,738 patches were IDC negative, while 78,786 were IDC positive. In this project, the information regarding which patient each patch belonged to was not considered as the models were only trained to identify whether each individual patch had IDC or not. It is also important to note the model only classified patches, not patients, and around 24,000 patches were used to train each model, so there was a sufficient amount of data for the model to train on. The dataset was organized as follows: 1) a patient number corresponds with one whole mount slide image, 2) within each patient's folder, the 50 x 50 patches are created, 3) based on whether or not a patch has IDC, it is sorted into folders of 0 (IDC negative) and 1 (IDC positive).

The patches from only 50 patients, as opposed to the complete 162, were used in this project due to time and computational constraints. Simply uploading the patches into Google Drive took hours for each patient, which was a clear indication that the dataset was too large for the computational resources used in this project. The dataset was then split into a training set, a validation set, and a testing set. The patches from 40 patients were used for training, 5 for validation, and 5 for testing. However, in these subsets, there were class imbalances, meaning the two classes (IDC negative and IDC positive) were not equally represented. For the training set, there were nearly 3 times as many IDC negative patches as IDC positive. For the validation set, this imbalance was nearly 3.5, while for the testing set, the imbalance was more than 8 times.

In order to address the issue of class imbalance, downsampling was implemented. The number of IDC positive patches was smaller in comparison to IDC negative patches, so some IDC negative patches were removed. The IDC negative patches were randomly removed from the dataset until the two classes were roughly equal. This process was done for the training set, validation set, and testing set. Although downsampling is a concern in data science, it was suitable for this project as there was a sufficient amount of data for the models to learn from. In total, 21,607 images were used for training, 977 for validation, and 957 for testing. In each of these sets, the division of IDC negative and IDC positive patches was roughly even.

The Keras API was used to create the models because it is straightforward and high-level. Keras is also commonly used in the machine learning field. ResNet18 and ResNet34 are not included in Keras's list of available pre-trained models, so the library "Classification Models Zoo - Keras" from the Github user "qubvel" was imported into the program code.

First, the ResNet model was loaded in with the weights being set to "imagenet". For each ResNet model being tested (18, 34, 50, and 101), the respective model was loaded in. The "include_top" parameter was set to false so that the input shape would not be changed and the weights would be compatible. Next, the existing weights were set to not be trained because the model would be using ImageNet weights. Then, the layers were flattened to become a one-dimensional array of elements. After that, a model object was created with the number of classes being set to two (binary classification) and the activation set to sigmoid (used in binary classification). Finally, the loss was specified as binary cross-entropy, the optimizer was specified as "adam", and the metrics were specified. "Adam" was specified as the optimizer because it is commonly used. The metrics that were specified to be output by the program were accuracy, AUC (Area Under the ROC Curve), precision, recall, true positives, true negatives, false positives, and false negatives.

First, the data were pre-processed. This was done by rescaling the RGB values from 0-255 to 0-1 because typical values of 0-255 would be too high for the model to process. This was done for all images. Then, the file paths to the training, validation, and testing datasets were specified. Batch size, the number of samples passed through the model at a time, was specified to be 32. Then, the model was fit. Early stopping was used to determine the number of epochs to train for and was set to monitor validation loss so that when validation loss stopped improving, the model would stop training. The patience parameter in early stopping was set to five so that the model would stop training once the validation loss had stopped improving for five epochs. Early stopping was also used to prevent overfitting, which occurs when the predictive model fits exactly against its training data and is unable to properly predict unseen data. Finally, after training, the model was evaluated using the test set. The code for all four models was identical except for importing the ResNet model and creating an object with it. In this section of the code, it was modified to be either ResNet18, 34, 50, or 101.

All programs were run on Google Colab using the NVIDIA-SMI 470.42.01 GPU. The computer used during experimentation was a MacBook Pro on the operating system macOS Big Sur Version 11.4. Because this project was run solely on the computer, no safety precautions were needed or taken.

Results

The metrics used for evaluation in this project were F1-score, accuracy, precision, and recall. F1-score was used as the primary measure of performance as it is a common statistical analysis of binary classification. F1-score shows the balance between precision and recall. Precision finds the proportion of positive identifications that were actually correct, while recall finds the proportion of actual positives that were identified correctly. Accuracy can be described as the fraction of predictions the model got correct. Below is the formula for F1-score:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Figure 1: The formula for F1-score. The F1-score for each model during training, validation, and testing was calculated with the precision and recall scores output by the model.

The following table summarizes the F1-scores in training, validation, and testing for all four models (ResNet18, ResNet34, ResNet50, and ResNet101):

Set	ResNet18	ResNet34	ResNet50	ResNet101
Training	0.7103	0.7240	0.7369	0.7505
Validation	0.5384	0.5783	0.7127	0.7541
Testing	0.5816	0.6050	0.6967	0.7249

Figure 2: F1-scores for ResNet18, ResNet34, ResNet50, and ResNet101 in training, validation, and testing.

As seen in the above table, as the size of the model increases, the F1-score increases. This is true across training, validation, and testing. All of the models had a training F1-score between 0.71 and 0.76, while there was more variation in the validation and testing sets. The F1-scores in the validation set ranged from 0.5384 to 0.7541, with ResNet18 performing the worst and ResNet101 performing the best. Finally, in the testing set, F1-scores ranged from 0.5816 to 0.7249, with, once again, ResNet18 having the lowest performance and ResNet101 having the highest performance. Accuracy scores were not displayed in the table because they were extremely similar to the F1-scores. Across all sets and models, the accuracy score was only 0.001 greater than or less than the F1-scores.

Additionally, the training time of each model and the number of epochs each model trained for varied. The general trend of the training time for the model was that as the size of the model increased, the training time also increased, although there was an outlier, ResNet50. ResNet18 had a training time of approximately 53 minutes, while ResNet34 had a training time for about 77 minutes. ResNet50 had a training time of around 51 minutes, and ResNet101 trained for about 95 minutes. Because early stopping was implemented, the number of epochs each model trained for varied. ResNet18 trained for 13 epochs, ResNet34 trained for 17 epochs, ResNet50 trained for 12 epochs, and, finally, ResNet101 trained for 7 epochs.

The following graphs show the training accuracy and validation accuracy over the number of epochs for each model:

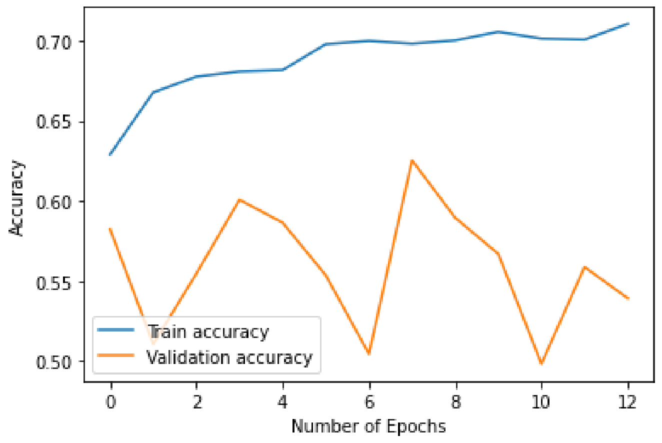

Figure 3: Training accuracy and validation accuracy of the model using ResNet18.

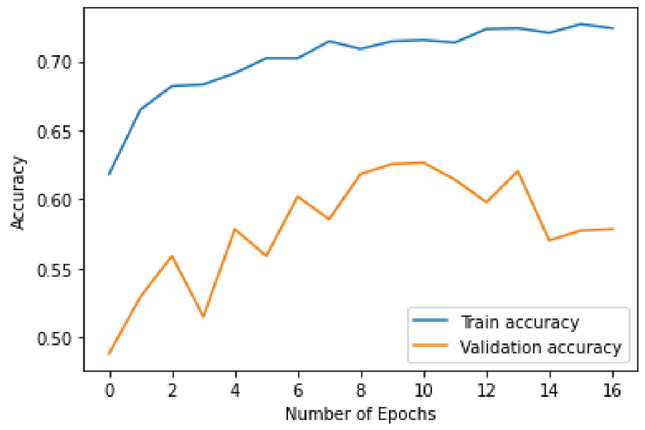

Figure 4: Training accuracy and validation accuracy of the model using ResNet34.

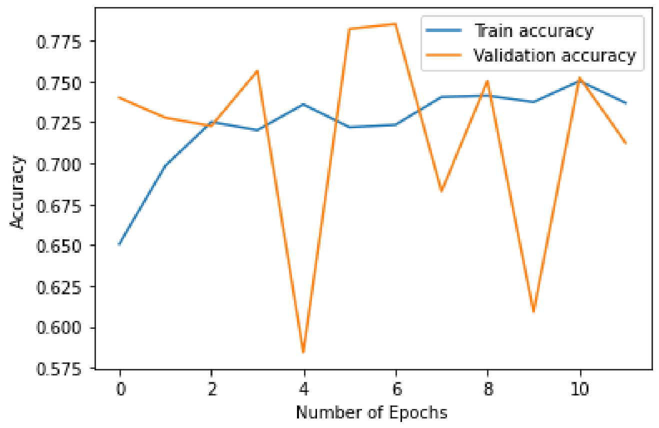

Figure 5: Training accuracy and validation accuracy of the model using ResNet50.

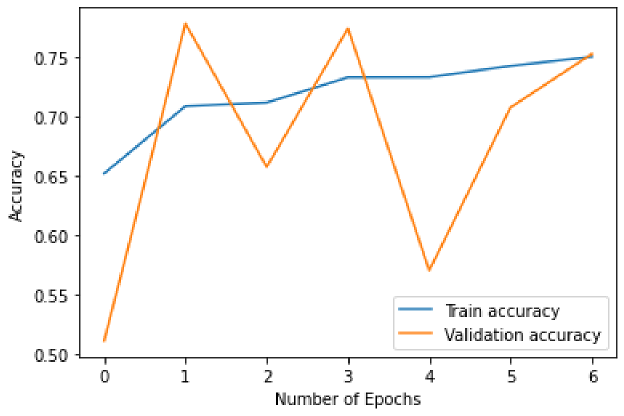

Figure 6: Training accuracy and validation accuracy of the model using ResNet101.

Discussion and Conclusions

The F1-scores from training, validation, and testing across all models support the hypothesis. It was hypothesized that as the size of the ResNet increases, the more accurate the model will be. The results of this experiment support this hypothesis because in the training set, validation set, and testing set, ResNet18 had the lowest F1-score, ResNet34 had the second lowest F1-score, ResNet50 had the second highest F1-score, and ResNet101 had the highest F1-score.

One interesting result from this experiment was the fluctuation of the validation accuracy across all models. Although the training accuracy had a general upwards trend and later stabilized after training for a certain number of epochs, the validation accuracy continued to spike up and down. In order to solve this issue, further research and fine-tuning of these models would be beneficial. Data augmentation, which applies transformations to the data such as horizontal flips, could also be implemented to introduce more variety into the images that the model trains on. More specific hyperparameter optimization could be investigated as well. Finally, increasing the patience in early stopping could be a potential solution to the validation fluctuation. Patience is a parameter in early stopping that determines when the model should stop training. If the validation loss does not change for the number of epochs specified by the patience, the model will stop training. Increasing the patience would increase the number of epochs the model trains for, which could be a way to reduce variability in the validation accuracy.

An interesting exploration for future research could be examining the impact of k-fold cross validation, a sampling procedure used to reduce bias in the dataset. Another interesting aspect of computer vision and transfer learning to research would be comparing the effect of different types of pre-trained models on their ability to recognize cancer. In addition to ResNet, models such as VGG and Inception could be compared.

In this paper, the effect of the ResNet's size on the performance of the model was explored. Four models using different sizes of ResNet (18, 34, 50, and 101) were trained and tested on a dataset of images with invasive ductal carcinoma. This paper attempted to address the current gap in knowledge regarding the role of a ResNet’s size in a model's performance in cancer identification. Although other papers apply convolutional neural networks of different sizes to cancer identification and classification, none of them examine the effect of a pre-trained model's size on its performance in recognizing breast cancer (Alanazi et al., 2021). Studying the performances of different-sized pre-trained models is important because data is difficult to access in healthcare, and pre-trained models reduce the amount of data needed to train models. Although the performance of the models is not yet ready for real world use, they open a multitude of computer vision and transfer learning applications in healthcare, especially in a field such as oncology.

Literature Cited

Peer-reviewed and Scholarly References

Alanazi, S. A., Kamruzzaman, M. M., Islam Sarker, M. N., Alruwaili, M., Alhwaiti, Y., Alshammari, N., & Siddiqi, M. H. (2021). Boosting breast cancer detection using convolutional neural network. Journal of Healthcare Engineering, 2021.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.

Cruz-Roa, A., Basavanhally, A., González, F., Gilmore, H., Feldman, M., Ganesan, S., ... & Madabhushi, A. (2014, March). Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In Medical Imaging 2014: Digital Pathology (Vol. 9041, p. 904103). SPIE.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Page, D. L. (2003). Special types of invasive breast cancer, with clinical implications. The American journal of surgical pathology, 27(6), 832-835.

Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., & Bray, F. (2021). Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA: a cancer journal for clinicians, 71(3), 209-249.