# CI7520 Assignment 2 – MedMNIST Classification

Group number: 9

Student names and k-numbers:

1. Megha Hegde - K2162274
2. Negar Abbasi - K2203561
3. Soniya Makarand Bhambure - K2203674
4. Prasanth Pradeep Kumar - K2238510

Table of Contents

[CI7520 Assignment 2 – MedMNIST Classification 1](#_Toc129886632)

[Introduction 2](#_Toc129886633)

[BloodMNIST 2](#_Toc129886634)

[Aims and Objectives 2](#_Toc129886635)

[Literature Review (analysis of existing work) 2](#_Toc129886636)

[Data Pre-Processing 3](#_Toc129886637)

[Transfer Learning using VGG16 4](#_Toc129886638)

[Original Models 5](#_Toc129886639)

[Original Model 1: Residual Blocks 5](#_Toc129886640)

[Original Model 2: Simple CNN, using Keras Tuner 6](#_Toc129886641)

[Comparative Analysis and Evaluation of Models 7](#_Toc129886642)

[BreastMNIST 10](#_Toc129886643)

[Aims and Objectives 10](#_Toc129886644)

[Literature Review (analysis of existing work) 10](#_Toc129886645)

[Data Pre-Processing. 11](#_Toc129886646)

[Model 1: Transfer learning using ResNet50 12](#_Toc129886647)

[Model 2: Original CNN 13](#_Toc129886648)

[Comparative Analysis and Evaluation of Models 13](#_Toc129886649)

[Website 15](#_Toc129886650)

[Blood Sample Classification 15](#_Toc129886651)

[Breast Sample Classification 15](#_Toc129886652)

[References 17](#_Toc129886653)

# Introduction

This project presents a deep learning approach for image classification in the BloodMNIST and BreastMNIST datasets. Firstly, the BloodMNIST datasets was explored, and this distribution of samples across the eight classes was identified. To standardize the dataset, the images were preprocessed by normalizing pixel values and resizing to a uniform size. Two techniques (class weights and SMOTE) were investigated to address the problem of imbalanced data distribution across the classes.

For the BloodMNIST dataset, which has 8 classes, we developed a novel deep learning model using CNN and residual blocks. We also applied transfer learning using VGG16.

Similarly, for the BreastMNIST dataset, which has two classes, malignant and benign, we developed deep learning models using CNN and transfer learning with ResNet50. We evaluated the performance of all models using metrics such as accuracy, precision, recall, and F1 score.

Our results show that the residual block model achieved the best performance for the BreastMNIST dataset with an accuracy of 92.81%, while the novel CNN model with Smote achieved the best performance for the BloodMNIST dataset with an accuracy of 82.69%.

Ultimately, our goal is to deploy these models for real-world classification of breast cancer and blood samples. We integrate these models into a web application for ease of use and accessibility.

# BloodMNIST

## Aims and Objectives

: The aim of this section is to develop a deep learning method for classification of images in the BloodMNIST dataset. To achieve this, the following objectives must be fulfilled

1. Explore the dataset, apply required pre-processing, and address the problem of imbalanced distribution of samples across the classes.
2. Review state-of-the-art deep learning models used for similar tasks.
3. Apply transfer learning using existing model(s) to the task at hand.
4. Develop a novel deep learning model for this task.
5. Evaluate all models using the same metrics and perform a comparative analysis to determine the best model.

## Literature Review (analysis of existing work)

Since the development of AlexNet by Krizhevsky et al. in 2012 [5], convolutional neural networks (CNNs) have exploded in popularity in many fields, and medical image classification is no exception.

Loey *et al*. [6] applied AlexNet to a binary classification task on blood microscope images, achieving 100% classification accuracy. This was achieved by performing transfer learning using an AlexNet model pre-trained on ImageNet and replacing the final three layers with layers appropriate for the classification task at hand.

VGG-16 was proposed by Simonyan and Zisserman [8] in 2014, as an improvement on AlexNet. VGG16 increased the depth of the network while reducing the size of the convolution filters. Yu *et al.* [12] showed that the increased depth of VGG16 meant that it was better at extracting discriminant information to classify images, compared to AlexNet.

VGG16 is still used in state-of-the-art solutions for medical image classification. A 2022 study by Abhishek *et al.* [7] implemented transfer learning using VGG16 for a binary classification task on blood microscope images. By fine-tuning the fully connected layers, and the last three convolutional layers, an accuracy of 96% was achieved.

Due to the high performance of VGG16 on similar tasks, the decision was made to implement transfer learning with this model for the BloodMNIST classification task.

## Data Pre-Processing

BloodMNIST is a dataset of 17,092 blood cell microscope images of eight different classes. Henceforth, the classes will be referred to by numbers as per Table 1, e.g., “basophil” will be referred to as class 0.

The dataset was split by the creators into training, validation, and test sets with a ratio of 7:1:2 [1]. The distribution of samples across classes in the wider dataset (Table 1) is reflected almost exactly across the training, validation, and test sets, hence the original split was used for the experiments in this paper.

|  |  |  |
| --- | --- | --- |
| Class Number | Class Name | Percentage of samples in dataset |
| 0 | basophil | 7.1 |
| 1 | eosinophil | 18.2 |
| 2 | erythroblast | 9.1 |
| 3 | immature granulocytes (myelocytes, metamyelocytes and promyelocytes) | 16.9 |
| 4 | lymphocyte | 7.1 |
| 5 | monocyte | 8.3 |
| 6 | neutrophil | 19.5 |
| 7 | platelet | 13.7 |

Table : Percentage of samples in each class in the BloodMNIST dataset.

As shown in Table 1, the dataset is imbalanced, with some classes having far more samples than others. Two methods were tested to address this issue, and the results were compared against each other and the original, unbalanced, dataset. To compare the methods, the VGG16 model was run three times with the same hyperparameters, with the only difference being the method of tackling data imbalance.

For the first method, class weights were calculated, and applied to the model at training time. Including class weights provided slightly more consistent accuracy across the classes, but classes 0 and 3 still had poor accuracy compared to the others (Figure 1). This technique reduced the overall prediction accuracy from 91.7% to 90.97%.

A popular method of addressing data imbalance is sampling, the fundamental principle of which is to alter the distribution of training samples. The second method tested for the BloodMNIST classification task was SMOTE (Synthetic Minority Oversampling Technique) [9], which is an oversampling technique that creates synthetic samples for minority classes. SMOTE creates a synthetic sample for a given class by using linear interpolation to combine two near-positive samples of the class [10]. Resampling the training and validation sets using SMOTE improved the accuracy of the transfer learning model to 91.76%. As shown in Figure 1, it also provided much more consistent accuracy across the classes. As SMOTE demonstrated the best performance, it was decided to use this method for all models going ahead.

Graphical user interface, application, Teams

Description automatically generated

Figure : Confusion matrix for VGG16 transfer learning model for the original dataset (left), with class weights applied (middle), and with SMOTE applied (right). The values on the diagonal provide the percentage probability with which each class was correctly predicted. SMOTE provides a much more consistent prediction accuracy across classes, whereas class weights and the imbalanced dataset have quite different accuracies for each class.

The creators of the BloodMNIST dataset performed some pre-processing on the dataset. The source images were centre-cropped and resized to shape 28 x 28 x 3 (RGB images with height and width 28 pixels) [1].

Two additional preprocessing steps were completed for this task. Firstly, the image arrays were divided by 255, hence converting the pixel values to the range [0, 1]. The smaller pixel values reduce the computational cost associated with processing the image. Secondly, the images were resized to 224x224, the default input size for the VGG16 model, to improve performance while using VGG16. These preprocessing steps were implemented by adding Keras Rescaling and Resizing layers at the top of each model. This was done for each model implemented, to enable like-for-like comparison between the models. Incorporating the pre-processing steps into the models themselves enables these models to be used for making predictions from different datasets with minimal pre-processing required.

## Transfer Learning using VGG16

#### Model Architecture

Transfer learning was implemented using a VGG16 model pre-trained on the ImageNet dataset. The pre-trained model and weights were loaded in Keras, excluding the “top” of the model. All convolutional layers were frozen, except for the last two, which were trainable for fine-tuning.

A *Sequential* model was created in Keras. The first two layers were the Resizing and Rescaling layers mentioned in the data pre-processing section. Then, the pre-trained VGG16 model was added. After this, a *Flatten* layer was used to make the data one-dimensional before passing it to the next layer, which was a fully-connected layer with 256 units and a ReLU activation function. L2 regularisation was used in this layer to reduce the risk of overfitting [14]. A Dropout layer was included after this to further combat overfitting. The final layer was a Dense layer with 8 units and a softmax activation; this produced the output of the 8-class classification task.

#### Training Process

As mentioned in the data pre-processing section, this model was run three times with the same hyperparameters to compare the performance of the imbalanced dataset, applying class weights, and applying SMOTE.

The conditions used for these three experiments are as follows.

The Adam optimizer was used, with an initial learning rate of 1 x 10-4. A callback was used to reduce the learning rate in the case of the validation loss plateauing for 5 epochs, with the minimum learning rate allowed being 1e-6. The sparse categorical cross-entropy loss function was used, as the task was a multi-class classification with integer labels. The number of epochs was set to 25, with the option for early stopping if the validation loss plateaued for over 5 epochs, and the learning rate could not be further decreased. The sparse categorical cross-entropy loss was used for training, as this is a multi-class classification ask with integer labels.

## Original Models

Two novel deep learning models were proposed for this task. The following section details the model architectures and training processes.

### Original Model 1: Residual Blocks

#### Model Architecture

This model uses the residual block idea proposed in 2015 with the ResNet family of models [3]. This idea was proposed to tackle the “vanishing gradients” problem in backpropagation, whereby errors propagated through multiple layers eventually overwhelm gradient information. Residual connections provide a “shortcut” for gradient information to flow around noisy layers such as Dropout, hence allowing a noise-free version of the information to be passed to the next input [4].

It was decided to use the residual block code in Chollet (2021) [4] to produce a model similar to those in the ResNet family, but with a lower complexity in order to reduce training time and reduce the risk of overfitting on the small dataset at hand.

As per the previous model, the first two layers were the Resizing and Rescaling layers for pre-processing, and the final layer was a fully-connected (Dense) layer with 8 outputs and a softmax activation.

#### Training Process

The version of the training and validation sets resampled using SMOTE were used in the training process.

The hyperparameters for this model were tuned by trial and error. The best performance was provided by an *RMSprop* optimizer with a learning rate of 1 x 10-4 and momentum 0.7. The momentum term was introduced to prevent the optimizer from getting stuck in a local minimum or maximum, instead ensuring that it was able to converge to the best possible loss and accuracy values. The sparse categorical cross-entropy loss was used once again.

The number of epochs was set to 35, and an early stopping callback was used to ensure the model was not trained for longer than necessary. The callback would stop training if the validation loss showed no improvement for 5 epochs. The training hence stopped after 29 epochs.

### Original Model 2: Simple CNN, using Keras Tuner

#### Model Architecture

The hyperparameters of the CNN, such as the number of filters, kernel size, activation function, and dropout rate, were optimized using Keras Tuner. Keras Tuner is a hyperparameter tuning package for Keras which allows the user to automatically improve the hyperparameters of a given deep learning model. [11]

The model consists of three convolutional layers with 96, 64, and 32 filters after the scaling layers. To reduce the output's dimensionality, a max pooling layer is added after each convolutional layer. These layers are used to find features in images that are essential for classification. After the convolutional and pooling layers, the model has a flattening layer that turns the output of the convolutional layers into a 1D array. The data transfer from this layer to the following layers is simplified. A dropout layer with a dropout rate of 0.3 follows the flattening layer. This layer is used to reduce overfitting and improve the model's generalization performance. There are also two dense layers of 256 and 8 units each, having softmax activation in the final layer.

The Adam optimizer and sparse categorical cross-entropy loss are used to build the model.

#### Training Process

The search method in Keras Tuner was used to obtain the best model, the model with the combination of hyperparameters providing the best performance. Using the training and validation sets resampled by SMOTE, the model was trained for 20 epochs. During the training, the model achieved a validation accuracy of 86%.

The plots demonstrate the changes in training and validation accuracy and loss throughout the training period. It is observable that both training and validation accuracy increase and converge as the number of epochs increases. However, it seems that the validation loss increases while the training loss decreases, indicating a possible issue with overfitting.

A picture containing graphical user interface

Description automatically generated

Figure : Plots for accuracy (left) and loss (right) during training. The training and validation accuracy follow the same path over the course of training. However, the validation loss increases as the training loss decreases.

The model's accuracy on the test set is 89.86%, meaning that the model correctly classified 89.86% of the test set data. The accuracy, recall, and F1-score metrics are used to measure how well the model performed on each of the test classes. Each class's metrics are determined independently.

Class 1, 6, and 7 performed well, which can be seen by their respective f1-scores of 0.9863, 0.9555, and 0.9905, according to the evaluation measures. Classes 0, 3, and 5 had lower f1-scores, with respective values of 0.8306, 0.7665, and 0.7722.

## Comparative Analysis and Evaluation of Models

Table 2 provides metrics for the three final models, all using the training and validation sets resampled with SMOTE during training, and tested on the original test set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision (weighted avg.) | Recall  (weighted avg.) | F1 Score  (weighted avg.) |
| VGG16 | 91.76% | 91.77% | 91.76% | 0.9173 |
| Original Model 1 (Residual Blocks) | **92.81%** | **92.95%** | **92.81%** | **0.9287** |
| Original Model 2 (Simple CNN) | 89.86% | 90.46% | 89.86% | 0.8986 |

Table : Metrics for the three final models. These models all used the version of the training and validation datasets oversampled with SMOTE to tackle the data imbalance issue. Original Model 1 clearly had the best performance across all metrics shown here.

Accuracy refers to the percentage of images in the test dataset that were correctly classified (i.e. cases where the predicted label matched the true label).

The precision score for a class x refers to the proportion of cases where a sample is predicted as class x and is in fact class x. This is represented by the following equation [2]:

**Precision = Number of true positives/(Number of true positives + Number of false positives)**

The recall score for a class refers to the proportion of negative cases that were correctly classified, i.e.,

The recall score for a class x refers to the proportion of samples of class x that are correctly classified. This is represented by the following equation [2]:

**Recall = Number of true positives/(Number of true positives + Number of false negatives)**

The F1 score is a harmonic mean of the precision and recall, calculated using the following equation [13]:

**F1 Score = 2 x (precision x recall)/(precision + recall)**

The average and per-class precision, recall, and F1 score values for each model were calculated using the *classification\_report* method in the *scikit-learn* package. The values quoted in Table 2 are weighted averages. For instance, the weighted average precision is the mean precision across all classes in the dataset, weighted by the number of samples per class. Using weighted averages is therefore useful for cases like BloodMNIST where there is an imbalanced distribution of samples across classes.

The drawback of the F1 score is that it assigns equal weight to the precision and recall, whereas in real-life scenarios, they are not always equally important [13]. For medical image classification problems, it is it is crucial to correctly classify as many samples as possible, hence the recall is particularly important. Hence, accuracy and recall will be the metrics that are mainly focused on to compare the models for the BloodMNIST classification task.

Original Model 1 (the model produced using residual blocks) clearly performs best across all metrics measured reported in Table 2, and hence can be said to be the best model for the task.

Figure 2 shows the confusion matrices for the three final models. Each value in the confusion matrix represents a different true vs. predicted label case; for example, the value in the top left-hand corner represents the proportion of cases where the true label and the predicted label were both class 0. The values on the diagonal hence represent correct classifications, i.e., cases where the true label and the predicted label are the same. Comparing the confusion matrices for the models makes it less obvious which is the best model. Original Model 2 clearly shows the poorest performance; it has particularly poor performance on class 5, and often misclassifies samples of class 3 as class 5. However, the performance for VGG16 and Original Model 1 looks similar from the confusion matrices. They both have high values on the diagonals and lower values elsewhere. This shows the importance of looking at multiple metrics and visualisations to assess model performance.

Application, Teams

Description automatically generated

Figure : Confusion matrices for the three final models. The values on the diagonals represent correct classifications, e.g., having 0.91 in the top left-hand corner means that 84% of the time, class 0 was correctly predicted as class 0.

An important point that can be seen from the confusion matrices (Figure 2), and the per-class metrics for each model (Table 3-5), is that all the models seem to have poor performance on classes 3 and 5 compared to the other classes. The data imbalance has already been addressed via SMOTE, and additionally, class 3 is not a minority class – hence, there must be another issue.

It appears from the confusion matrices that classes 3 and 5 are sometimes misclassified as one another, suggesting that the appearance of the two classes may be similar, i.e., images of monocytes (class 5) and immature granulocytes (class 3) may look similar. To understand how to solve this problem, it is necessary to develop a further understanding of the dataset and the nature of the different classes. It may make sense to combine classes 3 and 5 if they are very similar, but this should only be done if they can in fact be grouped together in a way that makes sense in the context of blood microscopy. This is proposed as an area for further work.

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score |
| 0 | 0.9079 | 0.8484 | 0.8771 |
| 1 | 0.9339 | 0.9744 | 0.9537 |
| 2 | 0.9428 | 0.9003 | 0.9211 |
| 3 | 0.8293 | 0.8307 | 0.8300 |
| 4 | 0.8792 | 0.9588 | 0.9173 |
| 5 | 0.8370 | 0.8134 | 0.8250 |
| 6 | 0.9678 | 0.9489 | 0.9583 |
| 7 | 0.9915 | 0.9936 | 0.9926 |

Table : Per-class metrics for VGG16 transfer learning model

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score |
| 0 | 0.8434 | 0.8607 | 0.8519 |
| 1 | 0.9951 | 0.9840 | 0.9895 |
| 2 | 0.9258 | 0.9228 | 0.9243 |
| 3 | 0.8135 | 0.8515 | 0.8321 |
| 4 | 0.9740 | 0.9259 | 0.9494 |
| 5 | 0.8481 | 0.8451 | 0.8466 |
| 6 | 0.9800 | 0.9565 | 0.9681 |
| 7 | 0.9874 | 0.9979 | 0.9926 |

Table : Per-class metrics for Original Model 1 (residual blocks)

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score |
| 0 | 0.8375 | 0.8238 | 0.8306 |
| 1 | 0.9887 | 0.9840 | 0.9863 |
| 2 | 0.9500 | 0.8553 | 0.9002 |
| 3 | 0.8427 | 0.7029 | 0.7665 |
| 4 | 0.8085 | 0.9383 | 0.8686 |
| 5 | 0.6755 | 0.9014 | 0.7722 |
| 6 | 0.9605 | 0.9505 | 0.9555 |
| 7 | 0.9832 | 0.9979 | 0.9905 |

Table : Per-class metrics for Original Model 2 (simple CNN, using Keras Tuner)

# BreastMNIST

## Aims and Objectives

The BreastMNIST dataset is a freely available dataset that consists of digitized images of breast histopathology slides. With this dataset, machine learning models may be created that correctly categorize breast cancer in histopathology pictures. The objective of this project is to create a website with deep learning models for categorizing breast cancer using the BreastMNIST dataset.

* Data preprocessing: For ResNet, the first aim is to preprocess the dataset to prepare it for use with the ResNet50 model. This will involve resizing the images to 224x224 pixels and converting them to RGB format. For CNN greyscale images from the dataset will be working fine.
* Model development: The second aim is to develop a CNN and ResNet50 model for breast cancer classification. The model will be trained on the preprocessed dataset using a supervised learning approach. The model will be evaluated on a validation dataset to determine its accuracy and performance.
* Model optimization: The third aim is to optimize the ResNet50 model for breast cancer classification. This will involve fine-tuning the model by adjusting its hyperparameters to improve its accuracy and performance. Also, for CNN model SMOTE is used to get an optimized result.
* Model interpretation: The fourth aim is to interpret the ResNet50 model to gain insights into its decision-making process. This will involve using techniques such as saliency mapping and activation maximization to visualize the regions of the images that the model focuses on when making its predictions.
* Model deployment: The final aim is to deploy the ResNet50 and CNN model for breast cancer classification in a real-world setting. This will involve integrating the model into a web application or other software system that can be used by medical professionals to aid in breast cancer diagnosis.

This project's successful conclusion could result in a number of advantages. First, it will show that it is possible to classify breast cancer using histopathology photos and deep learning algorithms. Second, it will offer a trustworthy and accurate diagnostic tool for breast cancer that can be applied in many clinical situations. Finally, it will add to the expanding body of knowledge about the application of machine learning to medical image processing and could result in new advancements in this area.

## Literature Review (analysis of existing work)

In their 2016 paper, He et al. [3] introduced the ResNet architecture, including the ResNet-50 Model, which was tested on the ImageNet dataset, achieving state-of-the-art results. In 2018, Li and Hassanpour [18] used the ResNet-50 model for binary classification of breast cancer histology images. The authors fine-tuned the pre-trained ResNet-50 model on a dataset of 277 images and achieved an accuracy of 92.5%.

A 2020 study by Al-Haija and Adebanjo provided an overview of the strengths and limitations of using ResNet-50 for breast cancer histology image analysis and demonstrated that it provided excellent results. Zhou et al. demonstrated in their 2019 study that transfer learning using ResNet-50 also achieved high accuracy [number?] on binary classification of renal tumours in histopathological images.

Overall, these publications show that the ResNet-50 model is useful for binary classification of medical pictures, such as histology images of breast cancer, CT images of pulmonary nodules.  When the model is fine-tuned using smaller medical image datasets, it has demonstrated strong performance and high levels of accuracy. Hence, it was decided to implement transfer learning using ResNet-50 for the BreastMNIST classification task.

## Data Pre-Processing.

BreastMnist dataset from Zenodo has 780 ultrasound images of two different classes, **Malignant** and **Benign**. The datasets were split by the author into training, validation, and test sets with a ratio of 7:1:2. The training set was 546 (70%), test set was 156 (20%), and validation was 78 (10%). Various data preprocessing steps were carried out in order to train a CNN and ResNet50 model using the BreastMNIST dataset.

As its image classification preprocessing of images is important. Initially the image has been normalized by dividing the train, test, and validation images by 255.0 so the values range between 0,1. Further the image is formatted to (28,28,1) shape. The outcome of above is used only in CNN model with BreatMNIST dataset. The data has been further preprocessed by applying image **augmentation** on a dataset before sending it to a neural network, which gives the model a wider and more varied set of training instances, which will help it perform better and be more robust.

The model can learn to recognize features that are resistant to these modifications by applying various transformations to the photos, increasing the likelihood that it will generalize well to fresh, untried images. These alterations may take several forms, such as **adjustments** to the brightness, contrast, rotation, flipping, cropping, and zooming.

For CNN we have used the SMOTE method to handle the imbalanced dataset. **SMOTE** finds the k closest neighbors of a minority class sample it has chosen. Then, along the line segments linking the minority class sample to each of its k closest neighbors, fresh synthetic samples are generated. By picking a point at random along each of these line segments, the features of the two samples are combined to create the new synthetic samples. This procedure can be repeated for numerous minority class samples, allowing SMOTE to produce a balanced dataset with a target variable distribution that is more accurate [17]. After performing SMOTE distribution is checked whether they are balanced and move to next process.

The BreastMNIST dataset consists of grayscale images of size 28x28 pixels. The first step in data preprocessing was to resize the images to a larger size. This was done to improve the quality of the data and make it easier for the model to learn from the images. In this case, the images were resized to 224x224 pixels, which is the standard input size for ResNet50.

Next, the images were converted to RGB format. ResNet50 is a convolutional neural network that is designed to work with RGB images. Therefore, it was necessary to convert the grayscale images in the BreastMNIST dataset to RGB format before they could be used with the model. This was done by creating a new array with shape (224, 224, 3) and copying the values from the grayscale image into the first channel of the RGB image.

The images were then normalized. Normalization is a standard preprocessing step that is used to ensure that the data has a mean of zero and a standard deviation of one. This helps to improve the performance of the model and make it more robust to changes in the input data. In this case, the images were normalized by subtracting the Mean of the dataset from each pixel and then dividing by the standard deviation. Finally, the data was split into training and validation sets. This is an important step in machine learning, as it allows us to evaluate the performance of the model on data that it has not seen before. In this case, the data was split into a training set and a validation set in a ratio of 80:20. The training set was used to train the model, while the validation set was used to evaluate its performance.

In summary, the data preprocessing steps that were carried out for the ResNet50 model using the BreastMNIST dataset included resizing the images to a larger size, converting them to RGB format, normalizing the data, and splitting it into training and validation sets. These steps helped to improve the quality of the data and ensure that it was suitable for use with the model. By carrying out these steps, we were able to train a ResNet50 model that achieved high accuracy in classifying breast histopathology images.

## Model 1: Transfer learning using ResNet50

#### Model Architecture

The ResNet50 model is composed of several layers of different types, including convolutional layers, batch normalization layers, and pooling layers. The model also includes skip connections, which allow it to learn more effective representations of the input data. The basic building block of the ResNet50 model is the residual block. A residual block is composed of two convolutional layers with a batch normalization layer and a ReLU activation function in between them. The output of the second convolutional layer is added to the input of the block through a skip connection, which allows the model to learn residual representations of the input data. The ResNet50 model consists of 50 layers in total, with most of the layers being composed of residual blocks. The first layer of the model is a convolutional layer with 64 filters and a kernel size of 7x7. This is followed by a max pooling layer with a pool size of 3x3 and a stride of 2. The next four layers are composed of residual blocks with varying numbers of filters and kernel sizes.

The final layer of the ResNet50 model is a fully connected layer with 1,000 output nodes, which is used for classification tasks. In our case, we modified the final layer to output only 2 nodes, one for benign and the other for malignant classification.

The ResNet50 model has several advantages over other CNN architectures, including the ability to learn very deep representations of images and the use of skip connections, which allow the model to learn more effective representations of the input data. It is also known for its high accuracy and has been used successfully in a variety of computer vision tasks, including image classification and object detection.

In summary, the ResNet50 model is a deep neural network architecture that is capable of learning very deep representations of images. It is composed of several layers of different types, including convolutional layers, batch normalization layers, and pooling layers. The model also includes skip connections, which allow it to learn more effective representations of the input data. The ResNet50 model has been shown to be highly effective in a variety of computer vision tasks, including image classification.

### Model 2: Original CNN

#### Model Architecture

Next major process is training the CNN model. For image recognition, object detection, and numerous other computer vision applications, convolutional neural networks (CNNs) are a potent deep learning technology. CNNs draw their inspiration from the biological design of the human visual system, which employs a hierarchical method to identify patterns [15].

Automatic learning and feature extraction from images are aspects of the CNN architecture. Convolutional, pooling, and fully connected layers are among the many layers that make up this system.

Every filter has a unique purpose, such as identifying edges, corners, or forms. The feature maps' spatial dimensionality is decreased using the pooling layer, which increases model efficiency and lowers the possibility of overfitting. Max-pooling, the most popular pooling technique, chooses the highest value possible inside a region [16].

The final layer in the CNN architecture, which is utilized to classify the image, is the fully connected layer. The classification operation, such as identifying the object in the image, is carried out using the output from the preceding layers.

#### Training Process

A Flatten layer was added to transform multi-dimensional data into a **single dimensional** dataset. A Dropout layer was added to avoid overfitting,which is common in neural network models. This model was trained twice with the same parameters - once using the original (unbalanced) training and validation datasets, and once using the training and validation datasets after resampling by SMOTE. This was done to assess whether SMOTE did in fact improve the performance of the model. As expected, Model’s performance and accuracy has been increased from 79% to **83%.** Similarly other evaluation metrics have been increased like Precision, Recall and F1 Score [13].

After training, the models were saved to .hdf5/.h5 files for later use with the web application.

## Comparative Analysis and Evaluation of Models

Table 6 shows a comparison of the metrics evaluated for the three models explored for the BreastMNIST classification task, when tested on the test dataset. The original CNN model developed, trained on data resampled using SMOTE, performs the best across all metrics, and performs significantly better than the other two models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision (weighted avg.) | Recall  (Weighted avg.) | F1 Score  (Weighted avg.) |
| CNN (with imbalanced dataset) | 78.85% | 79.92% | 96.51% | 0.8669 |
| CNN (using SMOTE) | **82.69%** | **85.23%** | **92.07%** | **0.885** |
| ResNet (using SMOTE) | 65.38% | 90.63% | 59.48% | 0.7157 |

Table 6: Evaluation metrics for each model explored for the BreastMNIST classification task. The CNN model (using training and validation sets resampled using SMOTE) had the best performance across each of these metrics.

# Website

To provide a user interface for the models discussed above, the Medical Image Classification Hub website was created. The website was developed using the streamlit package in Python, and currently runs on localhost.

To run the website, it is necessary to download the medmnist\_website.py file, along with the model weights files, and save them all in the same directory. A Python environment must then be created, and the following packages installed: streamlit, tensorflow, and keras. The website can then be run by opening the command shell, navigating to the directory containing the files, activating the Python environment, and running the following command “streamlit run medmnist\_website.py”.

The website consists of three pages: the home page, the blood sample classification page, and the breast scan classification page. It is possible to navigate between the pages using the menu in the left-hand sidebar. The home page simply provides a welcome message and prompts the user to select the correct page from the sidebar for the image they wish to classify.

### Blood Sample Classification

The blood sample classification page contains a file uploader object for the user to upload an image of a blood sample. The uploader accepts .jpg, .jpeg, and .png files. Images of any size are supported, as the images are resized after uploading if they are not the correct size for the model. The image is displayed on the website after uploading to allow the user to check what has been uploaded.

After uploading, and any required resizing, the image is converted to a NumPy array and passed to the model using the *predict ()* method, to predict the label. The model used here is Original Model 1, which provided the best performance on the test set, as discussed above.

The output of the model is an array containing the probability that the sample belongs to each of the eight classes. The maximum element of this array is selected as the class which the sample belongs to. This number is then converted to the corresponding name of the class. The name of the class is then output on the screen for the user.

Spinners and loading messages are displayed while the image is being opened and displayed, and while the model is running. This is to ensure that users have a constant update on the status and are not left wondering whether the website is working.

The final output is displayed as per Figure 4.

### Breast Sample Classification

The breast sample classification works similarly to the blood sample classification, accepting different image modalities and sizes. The model used for this task is the original CNN model developed for BreastMNIST classification.

Again, the class with the maximum probability is selected as the class which the sample belongs to, and the name of the class is output for the user. The final output is displayed as per Figure 5.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Final display after classifying a blood sample image. The predicted class is displayed in the green box.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Final display after classifying a breast scan sample image. The predicted class is displayed in the green box.

# References

[1] Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., Ni, B. (2023) 'MedMNIST v2 – A large-scale lightweight benchmark for 2D and 3D biomedical image classification', *Scientific Data*, 10, article number 41. Available at: <https://doi.org/10.1038/s41597-022-01721-8>

[2] Shmueli, B. (2019) *Multi-Class Metrics Made Simple, Part I: Precision and Recall.* Available at: <https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2> (Accessed: 15th March 2023).

[3] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). Available at: <https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html>

[4] Chollet, F. (2021) Deep Learning with Python. Shelter Island, New York: Manning Publications Co LLC. Available at: <https://learning.oreilly.com/library/view/deep-learning-with/9781617296864/Text/09.xhtml#heading_id_7> (Accessed: 8th March 2023)

[5] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), pp.84-90. Available at: <https://doi.org/10.1145/3065386>

[6] Loey, M., Naman, M., Zayed, H. (2020) Deep transfer learning in diagnosing leukemia in blood cells. *Computers*, 9(2), pp.29. Available at: <https://doi.org/10.3390/computers9020029>

[7] Abhishek, A., Jha, R.K., Sinha, R. and Jha, K. (2022) Automated classification of acute leukemia on a heterogeneous dataset using machine learning and deep learning techniques. *Biomedical Signal Processing and Control*, 72(B), article number: 103341. Available at: <https://doi.org/10.1016/j.bspc.2021.103341>

[8] Simonyan, K. and Zisserman, A. (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. Available at: <https://doi.org/10.48550/arXiv.1409.1556>

[9] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16, pp.321-357. Available at: <https://doi.org/10.1613/jair.953>

[10] K. Li, W. Zhang, Q. Lu and X. Fang, "An Improved SMOTE Imbalanced Data Classification Method Based on Support Degree," 2014 International Conference on Identification, Information and Knowledge in the Internet of Things, Beijing, China, 2014, pp. 34­-38, doi: 10.1109/IIKI.2014.14.

[11] O’Malley, T., Bursztein, E., Long, J., Chollet, F. *et al.* (2019) *KerasTuner.* Available at: <https://github.com/keras-team/keras-tuner> (Accessed: 10th March 2023).

[12] Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H. and Rui, Y., 2016, June. Visualizing and comparing AlexNet and VGG using deconvolutional layers. In *Proceedings of the 33rd International Conference on Machine Learning*.

[13] Shmueli, B. (2019) *Multi-Class Metrics Made Simple, Part II: the F1-score.* Available at: <https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1> (Accessed: 15th March 2023).

[14] Koppert-Anisimova, I. (2021) *Stay away from overfitting: L2-norm Regularization, Weight Decay and L1-norm Regularization techniques*. Available at: <https://medium.com/unpackai/stay-away-from-overfitting-l2-norm-regularization-weight-decay-and-l1-norm-regularization-795bbc5cf958> (Accessed: 10th March 2023).

[15] Goodfellow , I. and Bengio, Y. (2016) *Deep learning*, *Deep Learning*. Available at: <https://www.deeplearningbook.org/> (Accessed: March 8, 2023).

[16] LeCun, Y., Bengio, Y. and Hinton, G. (2015) *Deep learning*, *Nature News*. Nature Publishing Group. Available at: <https://www.nature.com/articles/nature14539> (Accessed: March 10, 2023).

[17] Chawla, N.V. *et al.* (2002) *Smote: Synthetic minority over-sampling technique*, *Journal of Artificial Intelligence Research*. Available at: <https://www.jair.org/index.php/jair/article/view/10302> (Accessed: March 12, 2023).

[18] Vesal, S., Ravikumar, N., Davari, A., Ellmann, S. and Maier, A. (2018) Classification of breast cancer histology images using transfer learning. In *Image Analysis and Recognition: 15th International Conference*, ICIAR 2018, Póvoa de Varzim, Portugal, June 27–29, 2018, Proceedings 15 (pp. 812-819). Springer International Publishing. Available at: <https://arxiv.org/pdf/1802.09424>

[19] Q. A. Al-Haija and A. Adebanjo (2020) Breast Cancer Diagnosis in Histopathological Images Using ResNet-50 Convolutional Neural Network. *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, Vancouver, BC, Canada, 2020, pp. 1-7, doi: 10.1109/IEMTRONICS51293.2020.9216455.

[20] Zhou, L., Zhang, Z., Chen, Y.C., Zhao, Z.Y., Yin, X.D. and Jiang, H.B., 2019. A deep learning-based radiomics model for differentiating benign and malignant renal tumors. *Translational oncology*, *12*(2), pp.292-300. Available at: <https://doi.org/10.1016/j.tranon.2018.10.012>