**Name: Sriram Srikanth | Student ID: A0276528R | NUSNET ID: E1132261 | CS5242 Project Report Oct 2023**

## 1. Background of the project, background of the dataset, the ML task and its impact to the world

The project provides scope to experiment about **two primary aspects** – training over **different subsets** of the dataset and the impact of **additional information such as masks and pre-training**. The task primarily revolves around the dataset which consists of 5 classes of WBC microscopic images– basophils, eosinophils, lymphocytes, monocytes, and neutrophils. The auxiliary datasets provided are pRCC and Camelyon16. These datasets are key in extracting features and patterns and apply the insights over the WBC dataset in the form of transfer learning. Given the diverse nature of the dataset, the ML task can be approached in multiple ways and experimented with in a variety of training processes. **pRCC is an unlabelled image dataset, camelyon16 is a binary classification task and WBC deals with multi-class (5-class) classification.**
On initial inspection of dataset, it could be noticed that the **cam-16 dataset is well-balanced** across the two classes (normal and tumor), but there is a huge data sample **imbalance across the classes in the WBC dataset**. The final solution should be a **robust neural network model** that would perform well for all classes despite the dataset imbalance.

The imbalanced WBC dataset poses several challenges. Bias in Performance: Models favor the majority class, resulting in poor performance for the minority class. Accuracy Misleading: High accuracy can be deceptive; models may miss minority class, making accuracy an unreliable metric.Loss of Valuable Information: Underrepresented classes lead to information loss, critical in some applications. Generalization Difficulty: Models struggle to generalize to new data and may overfit to the majority class, performing poorly in real-world imbalanced scenarios.

**White Blood Cells** are play a vital role in the body's defense against cancer. They contribute to immune system, can be harnessed for immunotherapy, and their counts and activity can provide valuable information for monitoring cancer and its treatment.
**Lymphocytes** play a central role in the immune response against cancer.
T Cells are involved in cell-mediated immunity and can recognize and attack cancer cells. Tumor-infiltrating lymphocytes (TILs) are T cells that have penetrated tumor tissue and can have a positive impact on the prognosis of some cancer patients.
B cells produce antibodies, which can target cancer cells. Monoclonal antibody therapies, such as Rituximab for lymphoma, work by targeting cancer cells with antibodies.
**Neutrophils**: Neutrophils are the most abundant type of WBC and are essential for the innate immune response. They can help in the initial response to cancer cells, but their role in cancer is complex and can either promote or inhibit tumor growth depending on the context.
**Monocytes**: Monocytes can mature into macrophages, which are phagocytic cells that can engulf and digest cancer cells. Macrophages can also be involved in presenting cancer cell antigens to T cells, helping to activate an immune response.
**Eosinophils**: Eosinophils are involved in combating parasitic infections and allergic reactions.
**Basophils**: Basophils are involved in allergic and immune responses.

Classifying white blood cells (WBCs) is vital in the medical field for multiple reasons. **WBC composition and numbers aid in diagnosing diseases**; abnormal counts can pinpoint underlying causes (e.g., neutrophils for bacterial infections, lymphocytes for viral or blood disorders). Monitoring WBCs over time **tracks patient health and treatment responses**. Specific WBC types are linked to particular infections, guiding treatment choices. Assessing WBCs helps evaluate the immune system, detecting deficiencies or overactivity (e.g., eosinophils for allergies). Further, **applying neural networks on classification of WBC** would aid in medical advances, enhancing the understanding of WBC structure and function based on the type and contributing to improved patient care and therapy.

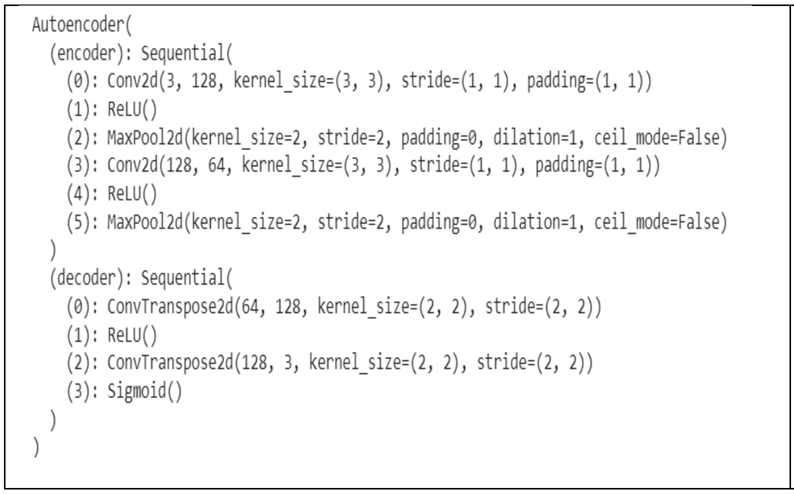
**For all of the datasets, the models were built from scratch.** I did not want the models to become too big of a black box and wanted to have a certain degree of control over the models, hence ready-to-load models such as VGG or ResNet-18 were **NOT** used.

## 2. pRCC

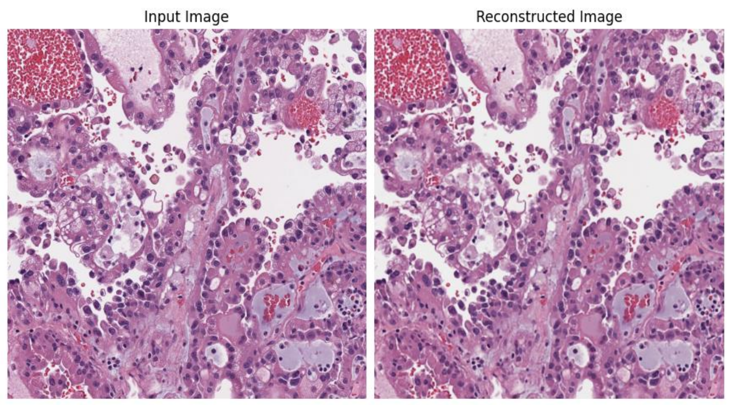
The pRCC dataset consisted of 384 * 384 dimensional images without labels, hence **unsupervised learning** seemed the best option.  To extract features from this dataset, an autoencoder was trained. Autoencoders have the ability to uncover crucial and abstract features in dataset. When these acquired features are applied to a different task, we essentially transfer adaptable knowledge. This allows the model to quickly adapt to the unique aspects of the new task while preserving valuable insights from the original task. **Autoencoders were chosen for this task, while also keeping in mind that the latent features should be used for pre-training WBC.**When we start the combined model with pre-trained autoencoder weights, it often converges to a satisfactory performance level more rapidly during fine-tuning. **These initial weights act as a solid foundation, reducing the amount of training time needed to achieve good results on the new task.**

In the encoder portion of the autoencoder, the input images is being subject to a series of convolutions that reduce the channels and dimensions. And in the decoder portion, the latent features are then reconstructed back to the original size.

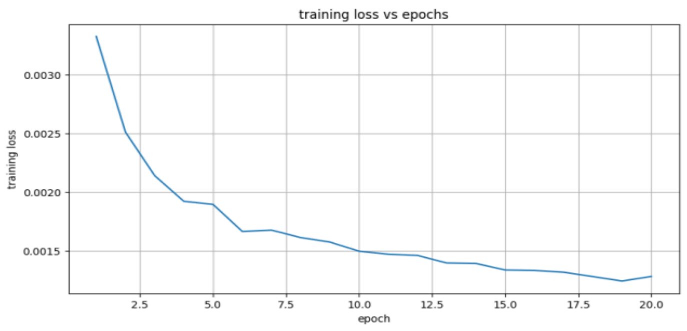
Final Validation Loss: 0.001165 (MSE). The autoencoder model was able to reconstruct the input images very well which implies that the model was also able to extract the latent features efficiently.



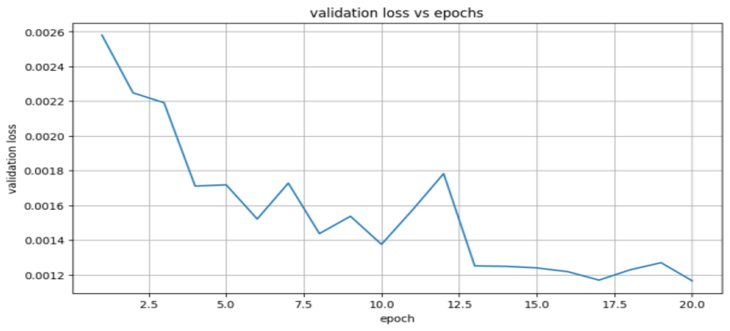
```
Autoencoder(
  (encoder): Sequential(
    (0): Conv2d(3, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(64, 128, kernel_size=(2, 2), stride=(2, 2))
    (1): ReLU()
    (2): ConvTranspose2d(128, 3, kernel_size=(2, 2), stride=(2, 2))
    (3): Sigmoid()
  )
)
```



Training Loss:



Validation Loss:



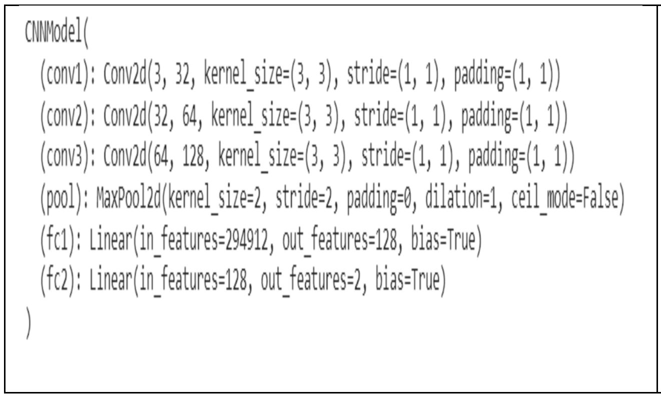
Code and Plots: pRCC.ipynb

### 3. Applying Masks to the images

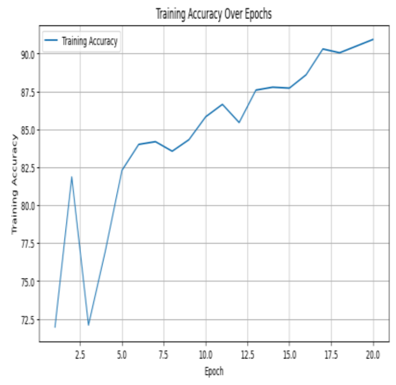
Masking images for the purpose of CNNs, serves as a powerful tool to concentrate on critical elements or regions in the input data while disregarding less important information. Masking allows the network to accentuate specific areas of interest, guiding its attention toward these regions. This is particularly valuable in this project like where the network must distinguish and prioritize objects within an image, improving its ability to recognize and focus on key features. The same mask code was used for cam-16 and WBC datasets, only the source and destination folder would differ.
Code: Mask.py

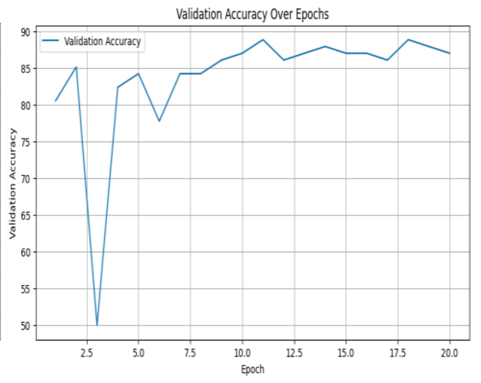
### 4. Cam-16

Given an image dataset with 2 classes - normal, and tumor, a binary classifier CNN would suit this task very well. The masks given along with the data were applied and the resulting images were also used with the original dataset.

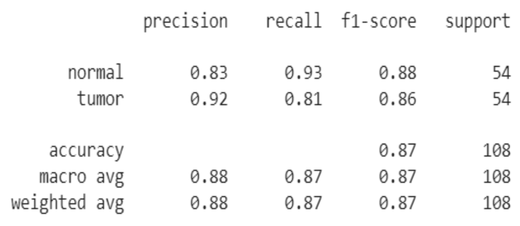


```
CNNModel(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=294912, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
```





Class-wise Precision, Recall and F1-Score:

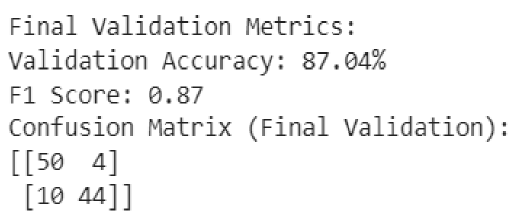


```
              precision    recall  f1-score   support

      normal       0.83      0.93      0.88        54
       tumor       0.92      0.81      0.86        54

    accuracy                           0.87       108
   macro avg       0.88      0.87      0.87       108
weighted avg       0.88      0.87      0.87       108
```

Overall validation score and Confusion Matrix:



```
Final Validation Metrics:
Validation Accuracy: 87.04%
F1 Score: 0.87
Confusion Matrix (Final Validation):
[[50  4]
 [10 44]]
```

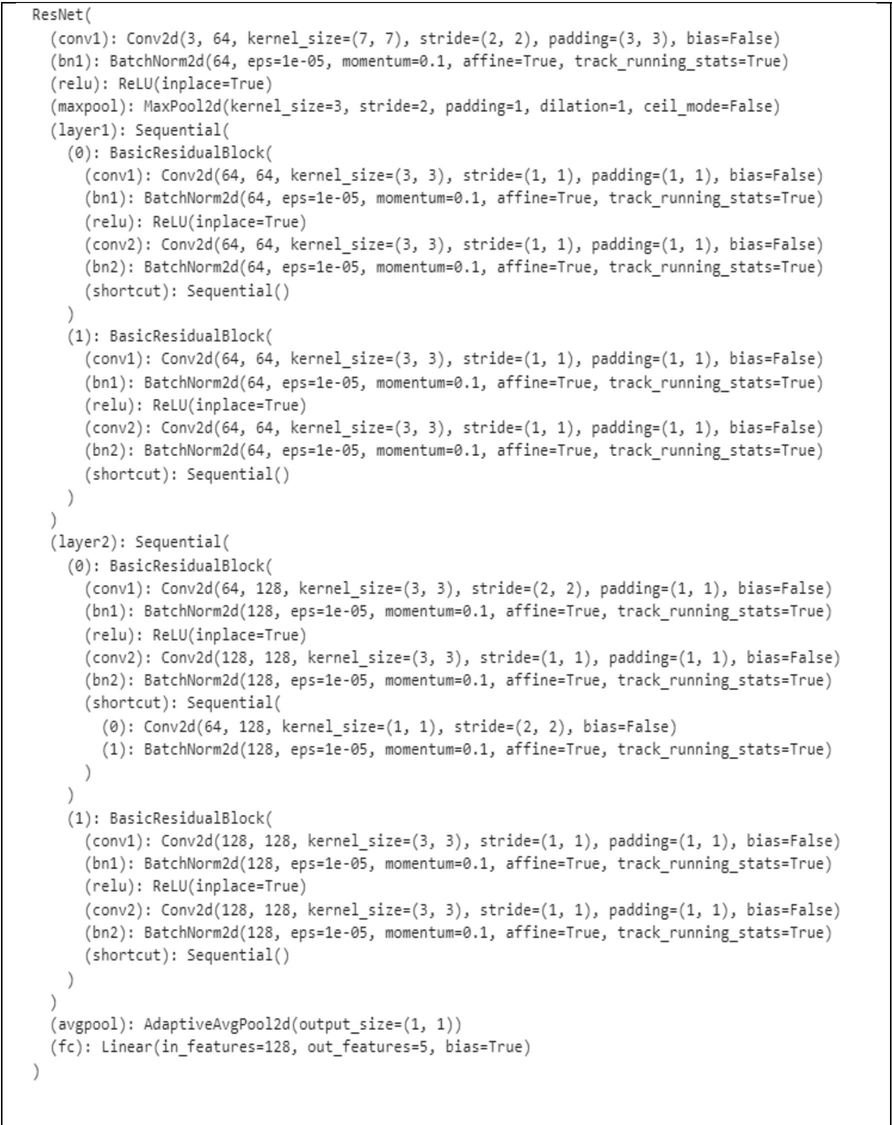
Code and Plots: camelyon.ipynb

**5. WBC (without any additional information):**

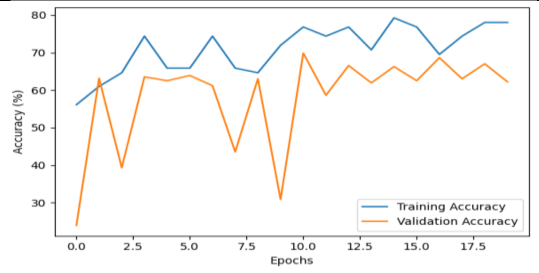
For the modelling of vanilla WBC, a CNN was designed. It consists of a combination of conv layers and residual layers. Using residual blocks along with conv layers would help to differentiate between 5 classes with good accuracy.
The architecture used was:

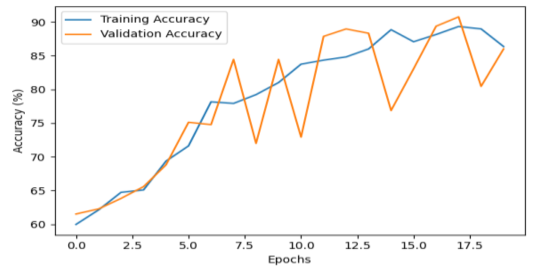
*Conv2d -> BatchNorm2d -> MaxPool2d ->ReLU ->Conv2d ->BatchNorm2D ->Residual block ->Residual Block ->Residual Block ->Residual Block -> Adaptive Average Pooling ->Linear*

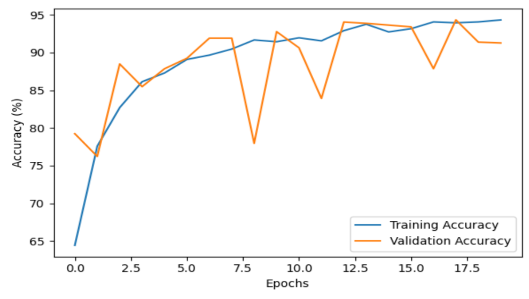
Detailed Architecture:

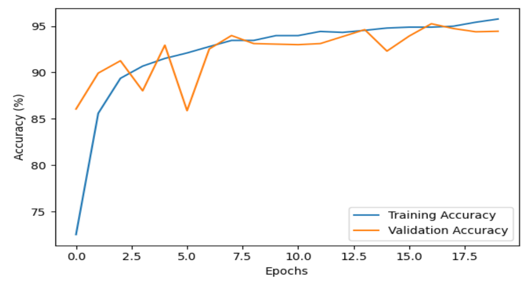


```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicResidualBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
    (1): BasicResidualBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (layer2): Sequential(
    (0): BasicResidualBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicResidualBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (shortcut): Sequential()
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=128, out_features=5, bias=True)
)
```









I have attached the **class-wise accuracy, precision, recall, f1 score and result interpretation of vanilla WBC in the next section** so that it can be compared side by side with WBC with additional information.

## 6. WBC with additional info (masks and pre-training with other models)

**WBC Using Masks:**

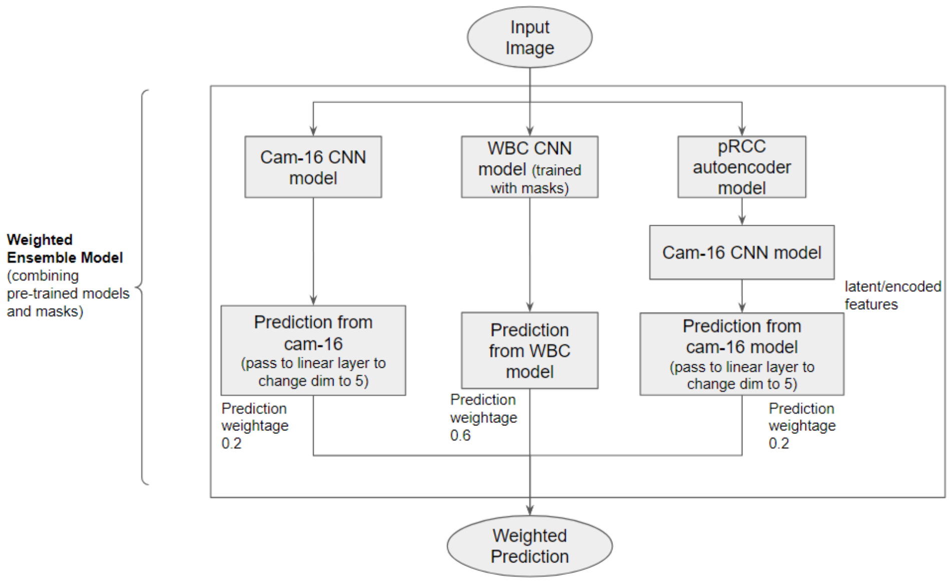
First, for each WBC dataset, using the same WBC architecture as before (mentioned in section 5) new models were trained but now also with the masks provided (masking code from mask.py). The given masks were applied to the corresponding images, and the resulting images were also stored with the original dataset and used for training.

The results of using masks and training WBC were better than the vanilla WBC model. Hence, an ensemble model was designed to get the most out of all models. This would combine the trained pRCC model, cam-16 along with the WBC models that were trained with masks on the WBC dataset.

**Final WBC Weighted Ensemble Model: pRCC + cam-16 + WBC using masks**

The ensemble model combines the models of pRCC, cam-16 and WBC (which was trained with masks)
Flow of the ensemble model:

- The **input image is fed to the cam-16 model**, which would have an output of size 2 (since it's a binary classifier). The results from cam-16 are then passed through a linear layer and converted into size 5.
  Let us call this **'prediction-a'**
- The **same input image is passed to the WBC model (that was trained with masks)** and the predictions are obtained. Let us call this **'prediction-b'**.
- The same input image is fed to the **encoder of pRCC to extract the latent features**. And the **encoded image is then passed to the cam-16 model** and as earlier, these are passed through a linear layer to obtain a prediction of size 5. Let us call this **'prediction-c'**.
- The **weighted ensemble model** would then combine the predictions in the following manner:
  ***weighted_prediction = ('prediction-a' * 0.2) + ('prediction-b * 0.6) + ('prediction-c' * 0.2)***


Input Image
Weighted Ensemble Model (combining pre-trained models and masks)
Cam-16 CNN model
WBC CNN model (trained with masks)
pRCC autoencoder model
Cam-16 CNN model
latent/encoded features
Prediction from cam-16 (pass to linear layer to change dim to 5)
Prediction from WBC model
Prediction from cam-16 model (pass to linear layer to change dim to 5)
Prediction weightage 0.2
Prediction weightage 0.6
Prediction weightage 0.2
Weighted Prediction

I wanted to provide higher weightage to the WBC model (trained with masks). Hence, I assigned a prediction weightage of 0.6 to that. This method led to improved results and also lesser number of epochs to achieve the improved results when compared to WBC (vanilla, no additional information).

Note: The WBC-1 model (trained with masks) was used in the ensemble model created for WBC-1. Similarly, the WBC-10 model (trained with masks) was used in the ensemble model created for WBC-10 and so on for WBC-50 and WBC-100.

It would not be a fair assessment to judge models based only the overall validation accuracy. Class-wise precision, recall and class-wise F1 scores are very important to bring the whole picture into context as we are dealing with a heavily imbalanced dataset. Hence class-wise metrics are also provided.

Code and Plots for content in the this page:
WBC-1_plain.ipynb, WBC-10_plain.ipynb, WBC-50_plain.ipynb, WBC-100_plain.ipynb, WBC_1_ensemble_with_masks_and_pretraining.ipynb, WBC_10_ensemble_with_masks_and_pretraining.ipynb, WBC_50_ensemble_with_masks_and_pretraining.ipynb, WBC_100_ensemble_with_masks_and_pretraining.ipynb

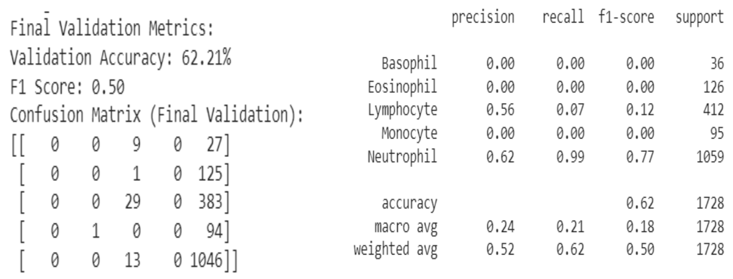
( Note: Confusion matrix format in the next page

|  | Predicted |  |  |  |  |
|---|---|---|---|---|---|
| Actual | basophil, | monocyte, | neutrophil, | lymphocyte, | eosinophil |
| basophil, | TP_1, | FP_1, | FP_2, | FP_3, | FP_4 |
| monocyte, | FN_1, | TP_2, | FP_1, | FP_2, | FP_3 |
| neutrophil, | FN_2, | FN_2, | TP_3, | FP_1, | FP_2 |
| lymphocyte, | FN_3, | FN_3, | FN_3, | TP_4, | FP_1 |
| eosinophil, | FN_4, | FN_4, | FN_4, | FN_4, | TP_5 |

)

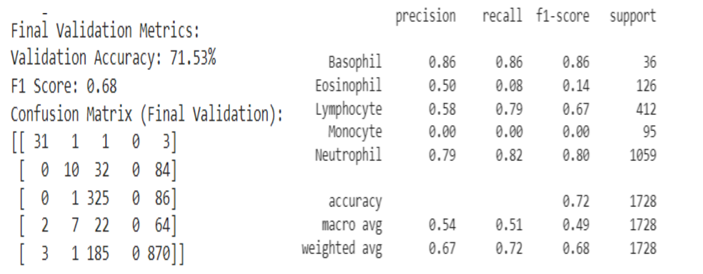
**Comparing WBC with and without additional info side by side:**
Class-wise Metrics: Precision, Recall, F1-Score
Final Metrics: Validation Accuracy, F1 scores, Confusion Matrix

**WBC-1: Without Additional Info (Vanilla WBC Model)**



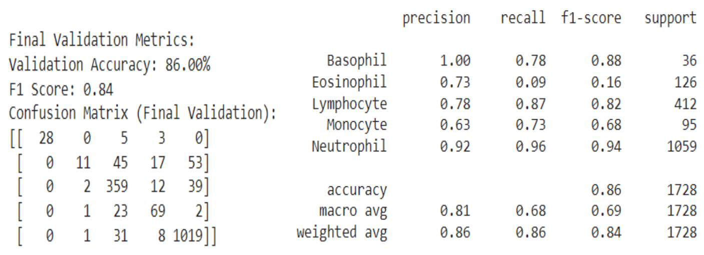
```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 62.21%
F1 Score: 0.50                          Basophil       0.00      0.00      0.00        36
Confusion Matrix (Final Validation):  Eosinophil       0.00      0.00      0.00       126
[[   0    0    9    0   27]           Lymphocyte       0.56      0.07      0.12       412
 [   0    0    1    0  125]             Monocyte       0.00      0.00      0.00        95
 [   0    0   29    0  383]           Neutrophil       0.62      0.99      0.77      1059
 [   0    1    0    0   94]
 [   0    0   13    0 1046]]            accuracy                           0.62      1728
                                       macro avg       0.24      0.21      0.18      1728
                                    weighted avg       0.52      0.62      0.50      1728
```

**With Additional Info (masks and pre-training)**



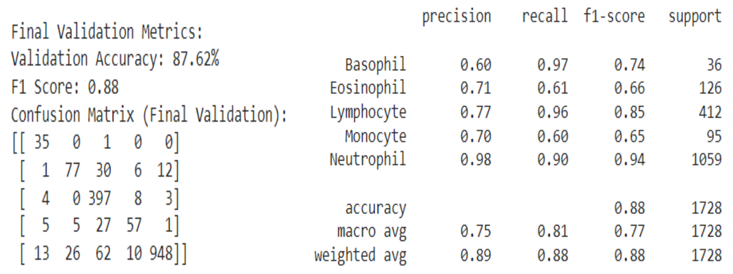
```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 71.53%
F1 Score: 0.68                          Basophil       0.86      0.86      0.86        36
Confusion Matrix (Final Validation):  Eosinophil       0.50      0.08      0.14       126
[[ 31   1   1   0   3]               Lymphocyte       0.58      0.79      0.67       412
 [  0  10  32   0  84]                 Monocyte       0.00      0.00      0.00        95
 [  0   1 325   0  86]               Neutrophil       0.79      0.82      0.80      1059
 [  2   7  22   0  64]
 [  3   1 185   0 870]]                accuracy                           0.72      1728
                                       macro avg       0.54      0.51      0.49      1728
                                    weighted avg       0.67      0.72      0.68      1728
```

**WBC-10: Without Additional Info**



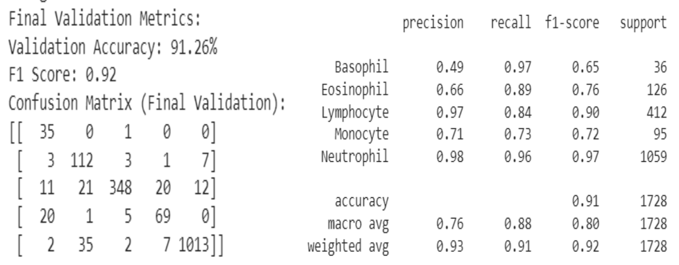
```
                                                  precision    recall  f1-score   support
Final Validation Metrics:
Validation Accuracy: 86.00%             Basophil       1.00      0.78      0.88        36
F1 Score: 0.84                        Eosinophil       0.73      0.09      0.16       126
Confusion Matrix (Final Validation):  Lymphocyte       0.78      0.87      0.82       412
[[  28    0    5    3    0]             Monocyte       0.63      0.73      0.68        95
 [   0   11   45   17   53]           Neutrophil       0.92      0.96      0.94      1059
 [   0    2  359   12   39]
 [   0    1   23   69    2]             accuracy                           0.86      1728
 [   0    1   31    8 1019]]           macro avg       0.81      0.68      0.69      1728
                                    weighted avg       0.86      0.86      0.84      1728
```

**With Additional Info (masks and pre-training)**



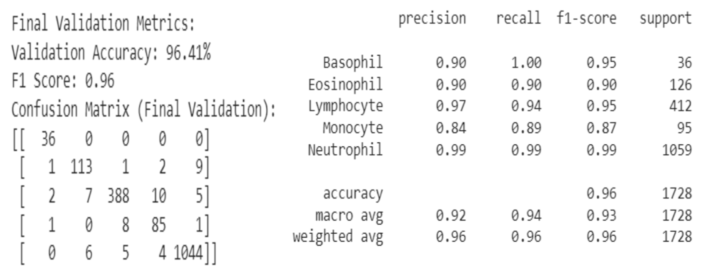
```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 87.62%
F1 Score: 0.88                          Basophil       0.60      0.97      0.74        36
Confusion Matrix (Final Validation):  Eosinophil       0.71      0.61      0.66       126
[[ 35   0   1   0   0]               Lymphocyte       0.77      0.96      0.85       412
 [  1  77  30   6  12]                 Monocyte       0.70      0.60      0.65        95
 [  4   0 397   8   3]               Neutrophil       0.98      0.90      0.94      1059
 [  5   5  27  57   1]
 [ 13  26  62  10 948]]                 accuracy                           0.88      1728
                                       macro avg       0.75      0.81      0.77      1728
                                    weighted avg       0.89      0.88      0.88      1728
```

**WBC-50: Without Additional Info**



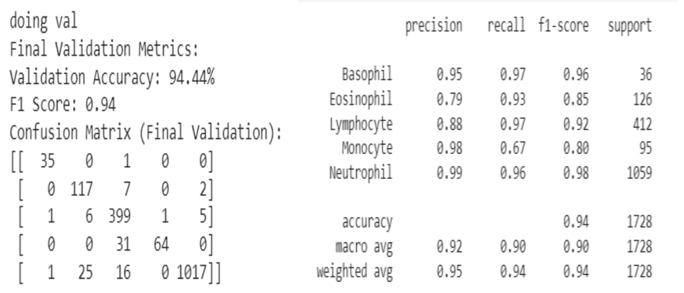
```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 91.26%
F1 Score: 0.92                          Basophil       0.49      0.97      0.65        36
Confusion Matrix (Final Validation):  Eosinophil       0.66      0.89      0.76       126
[[  35    0    1    0    0]           Lymphocyte       0.97      0.84      0.90       412
 [   3  112    3    1    7]             Monocyte       0.71      0.73      0.72        95
 [  11   21  348   20   12]           Neutrophil       0.98      0.96      0.97      1059
 [  20    1    5   69    0]
 [   2   35    2    7 1013]]            accuracy                           0.91      1728
                                       macro avg       0.76      0.88      0.80      1728
                                    weighted avg       0.93      0.91      0.92      1728
```

**With Additional Info (masks and pre-training)**



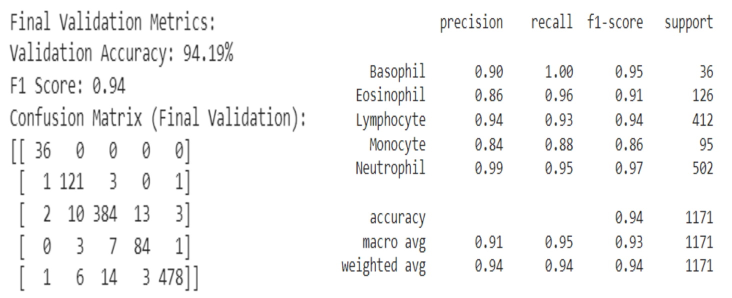
```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 96.41%
F1 Score: 0.96                          Basophil       0.90      1.00      0.95        36
Confusion Matrix (Final Validation):  Eosinophil       0.90      0.90      0.90       126
[[  36    0    0    0    0]           Lymphocyte       0.97      0.94      0.95       412
 [   1  113    1    2    9]             Monocyte       0.84      0.89      0.87        95
 [   2    7  388   10    5]           Neutrophil       0.99      0.99      0.99      1059
 [   1    0    8   85    1]
 [   0    6    5    4 1044]]            accuracy                           0.96      1728
                                       macro avg       0.92      0.94      0.93      1728
                                    weighted avg       0.96      0.96      0.96      1728
```

**WBC-100: Without Additional Info**



```
doing val                                         precision    recall  f1-score   support
Final Validation Metrics:
Validation Accuracy: 94.44%             Basophil       0.95      0.97      0.96        36
F1 Score: 0.94                        Eosinophil       0.79      0.93      0.85       126
Confusion Matrix (Final Validation):  Lymphocyte       0.88      0.97      0.92       412
[[  35    0    1    0    0]             Monocyte       0.98      0.67      0.80        95
 [   0  117    7    0    2]           Neutrophil       0.99      0.96      0.98      1059
 [   1    6  399    1    5]
 [   0    0   31   64    0]             accuracy                           0.94      1728
 [   1   25   16    0 1017]]           macro avg       0.92      0.90      0.90      1728
                                    weighted avg       0.95      0.94      0.94      1728
```

**With Additional Info (masks and pre-training)**



```
Final Validation Metrics:                         precision    recall  f1-score   support
Validation Accuracy: 94.19%
F1 Score: 0.94                          Basophil       0.90      1.00      0.95        36
Confusion Matrix (Final Validation):  Eosinophil       0.86      0.96      0.91       126
[[ 36   0   0   0   0]               Lymphocyte       0.94      0.93      0.94       412
 [  1 121   3   0   1]                 Monocyte       0.84      0.88      0.86        95
 [  2  10 384  13   3]               Neutrophil       0.99      0.95      0.97       502
 [  0   3   7  84   1]
 [  1   6  14   3 478]]                 accuracy                           0.94      1171
                                       macro avg       0.91      0.95      0.93      1171
                                    weighted avg       0.94      0.94      0.94      1171
```

The models that were trained with additional information, lead to better class-wise f1 scores and better overall performance. Across the data's subsets WBC-1, WBC-10, WBC-50, WBC-100 (in both with and without pre-training), we can see that the more data we train the model on, better is the result.  Getting more data for the model is giving it a chance to learn from many different and varied examples. This makes the model better at understanding new, unseen data. The model becomes better at figuring out the important patterns and connections in the data. Having more data also means the model can find and use more important details from the input data. This helps with the model to model figure out by itself how to represent the data with the underlying architecture.

And when we compare models that were trained on the same subset of dataset, but with and without additional info, it can be observed that the weighted ensemble model which uses additional info in the form of masks and pre-trained models pRCC, cam-16 has delivered better results overall and also class-wise - increased validation accuracy, f1 score and also better class-wiser performance. The pre-training has enabled the neural network to understand the dataset much better and distinguish the intricate details between each class of WBC.
Even for datasets such as WBC-1 and WBC-10 which do not have enough number of images, the weighted ensemble model was able to deliver good results class-wise. Pre-training has helped us to overcome to problems of dataset imbalance and data scarcity.

## 7. Conclusion:

pRCC is characterized by the uncontrolled growth of cells within the kidney tissue, while cancer metastases represent the spread of cancer cells from a primary tumor to other locations in the body, forming secondary tumors that retain the characteristics of the original cancer type. These secondary tumors consist of cancer cells from the primary tumor and are biologically similar to the primary tumor.

Pre-training using pRCC and cam-16 aided in improving the classification accuracy and f1 score of the models (overall and also class-wise). In the real world, we may not get enough volume of proper, structured data for our primary dataset for biomedical purposes. But if we do have good number of data (images) belonging to another part of the biomedical field which is similar to the dataset in which we are interested in, then we can pre-trained neural networks on the auxiliary data and the use transfer learning on top of the primary dataset. As seen in the case of WBC, pretraining leads to improved model performance.
While we observe the class-wise accuracy between without and with additional info models, We can see that the plain WBC model's WBC-1 and WBC-10 fail to perform well on Basophils, while I did fairly decent on the classes with enough samples.

**Basophils are uncommon WBCs. And hence we may not get enough number of images for those.** This can also be observed in the dataset. There is a huge class imbalance in the dataset, with Basophils comprising only a small portion of the dataset.  The number of images for the classes **Neutrophils and Lymphocytes are very high** compared to Basophils. Hence the vanilla model which was trained on the WBC dataset, was not able to perform well on this class. **But the proposed ensemble model was able to achieve good results even for classes which did not have enough number of data sample when compared to the other prominent classes.**

If we look at only the overall validation accuracy, it might seem like the WBC models without additional info are performing good. But since this is an imbalanced dataset, this would be a proper metric. To address these issues, it's important to consider other evaluation metrics such as class-wise precision, recall, F1-score when assessing the model's performance. These metrics provide a more comprehensive view of how well a model is performing, especially in situations where class imbalance or misclassification costs are a concern.

While analysing the structures of the classes, the similarities and differences across the 5 classes of WBC come into play. **Neutrophils** and **eosinophils** are granulocytes, which means they **have distinct granules** in their cytoplasm. Neutrophils are slightly smaller and have **multi-lobed nuclei** while **eosinophils have a bilobed nucleus** and are slightly larger. **Basophils**, another type of granulocyte, **have large, irregularly shaped granules** in their cytoplasm. They also have a bilobed nucleus. **Lymphocytes are typically smaller and round with a large, round nucleus** that occupies most of the cell. **Monocytes are the largest white blood cells with a large, kidney-shaped or horseshoe-shaped nucleus.**

The features learnt in the pRCC cam-16 pre-training are more general and transferable. This can lead to improved generalization to the WBC multi-class problem, as the models have already captured essential information about the data. With pretraining, we were able to overcome the class imbalance in the dataset and achieve good model performance for each class. Pre-training allows the model to adapt to the specific characteristics of the source domain, and then fine-tuning adapts it to the target domain. This is particularly useful when the source and target domains share some similarities, and in our case this was true.