White Blood Cell Classification

1605071 - Kazi Noshin

1605089 - Ishrat Jahan Eliza

Problem Definition

Classification of Four types of white blood cell:

1. Neutrophil
2. Eosinophil
3. Monocyte
4. Lymphocyte

Dataset

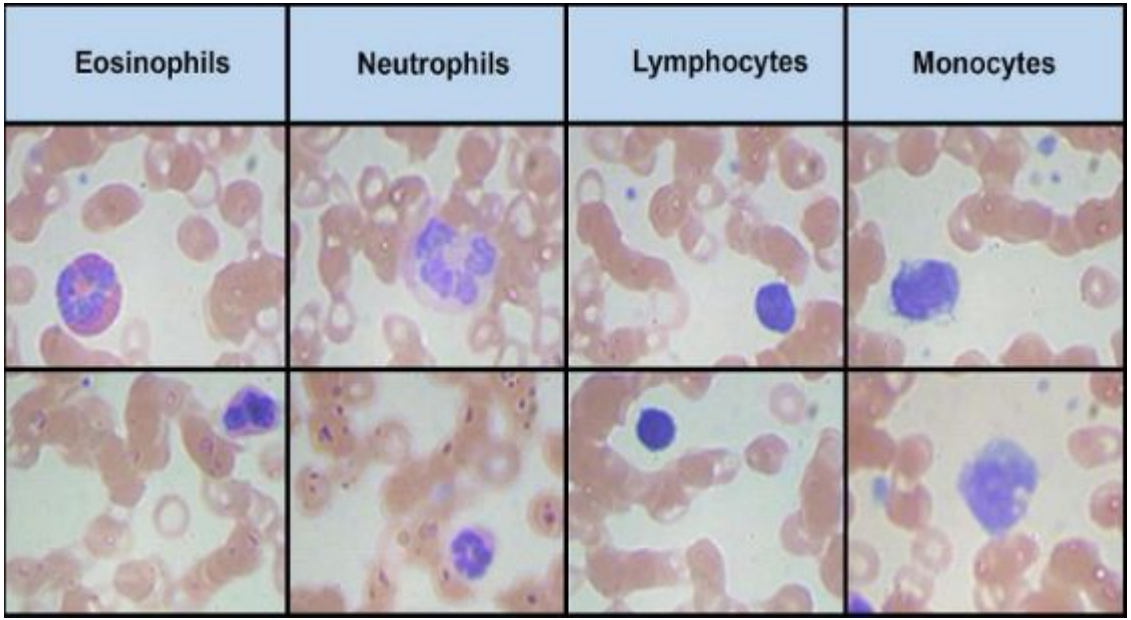
(from Kaggle) https://www.kaggle.com/paultimothymooney/blood-cells

Dataset type: Image

Dataset size:

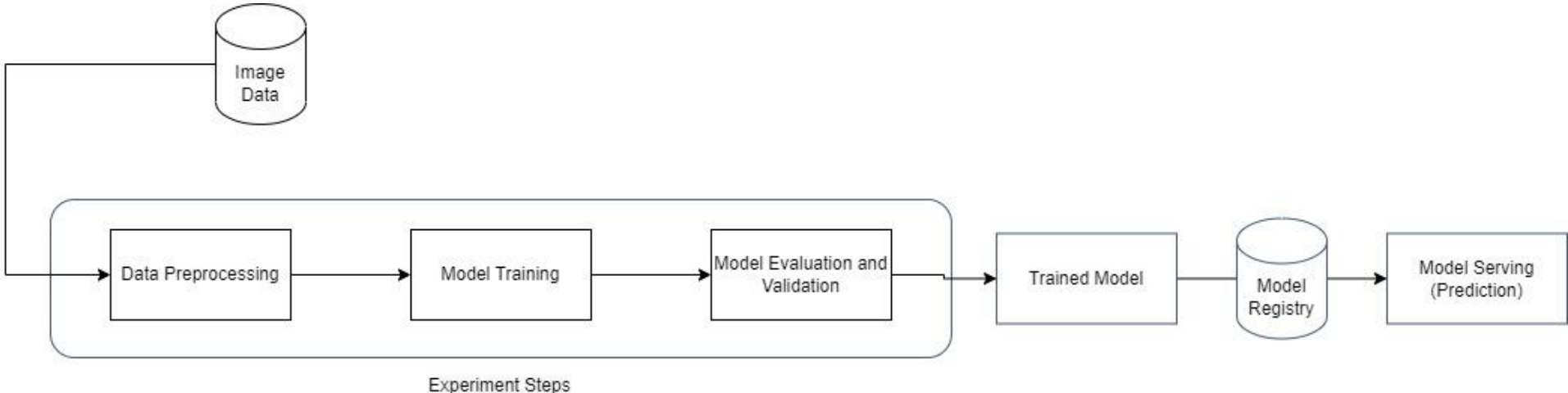
- 410 original image (pre-augmentation)
- 12,500 images in total (after augmentation)
- Approximately 3,000 images for each of 4 different WBC

White Blood Cells



Pipeline

- Dataset Preprocessing: (Balanced Dataset)
  - Stratified split of train set
  - Batch normalization (Because of ram shortage)
- Model Training
- Model Evaluation using Validation set



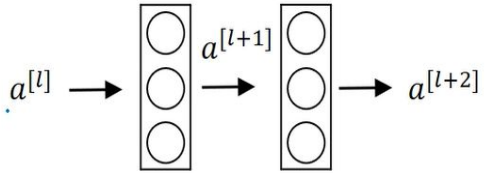
Architecture

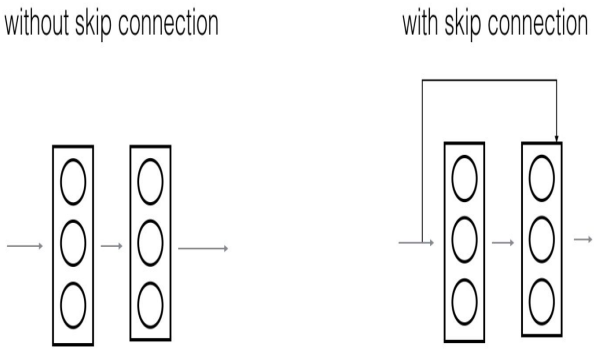
ResNet50

Framework: Keras

ResNet50

Skip connection (Residual block):





Without skip connection	**With** skip connection
Z[l+1] = W[l+1] * a [l] + b[l+1]	Z[l+1] = W[l+1] * a [l] + b[l+1]
a[l+1] = g(Z[l+1])	a[l+1] = g(Z[l+1])
Z[l+2] = W[l+2] * a [l+1] + b[l+2]	Z[l+2] = W[l+2] * a [l+1] + b[l+2]
a[l+2] = g(Z[l+2])	**a[l+2] = g(Z[l+2] + a[l])**

Skip Connection Image from DeepLearning.AI

ResNet50

Residual block:

- Identity Block
- Convolution Block

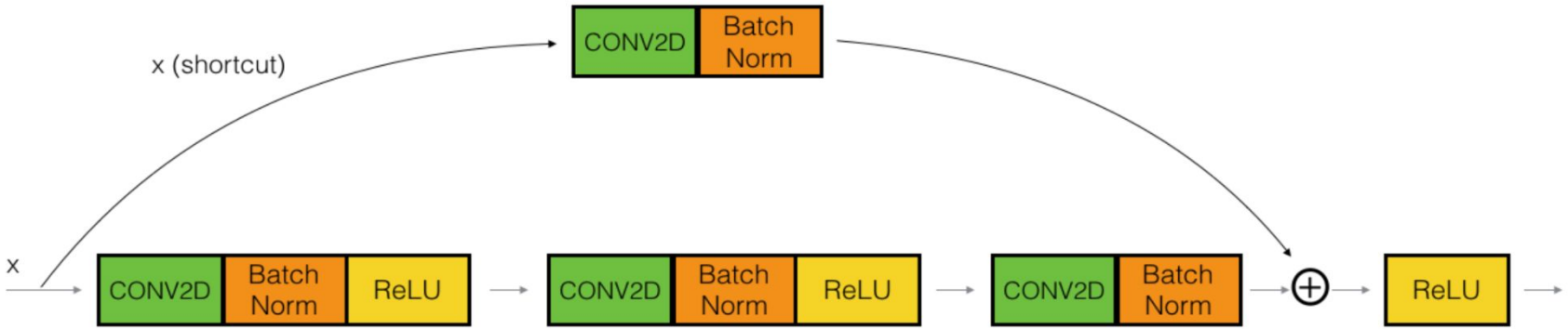
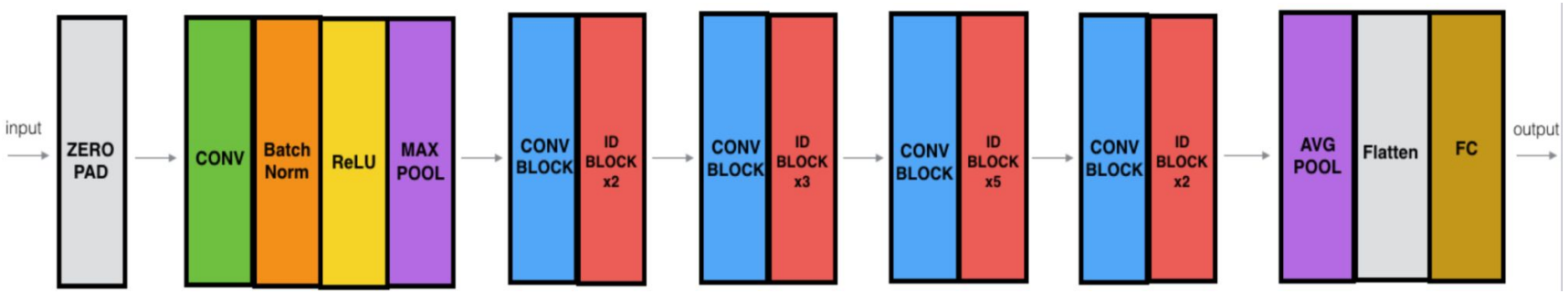


Fig: Convolution Block

ResNet50

Diagram:

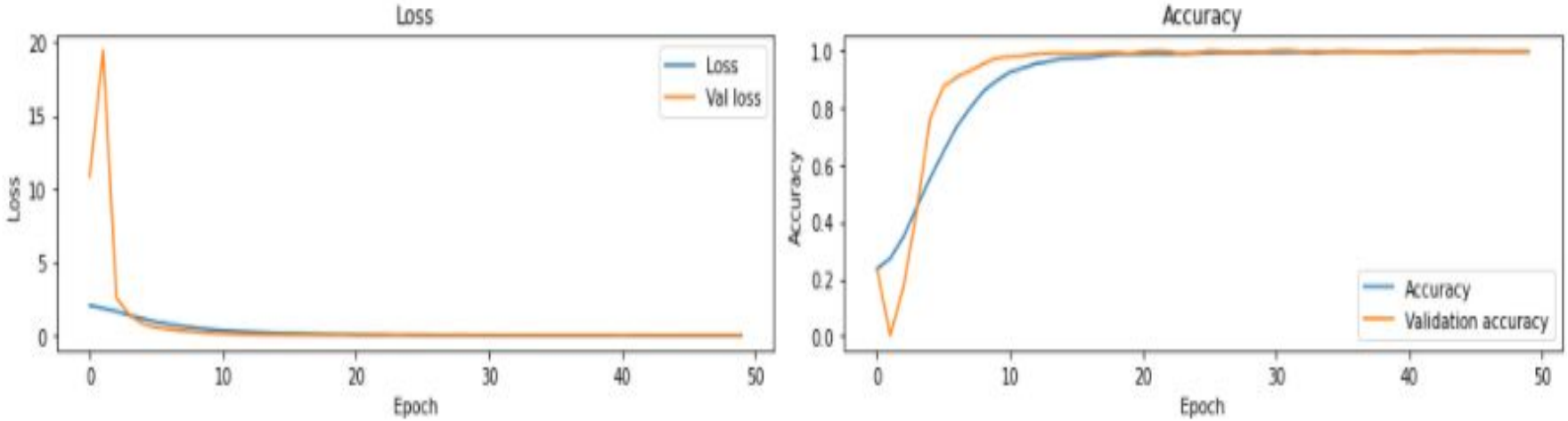


Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 4, 4, 2048)        23587712
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
batch_normalization (BatchNo (None, 32768)             131072
_________________________________________________________________
dense (Dense)                (None, 256)               8388864
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 128)               512
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
```

Performance

(transfer learning, dropout and L2 regularization, optimal epoch 25-30, Optimizer: Adam)

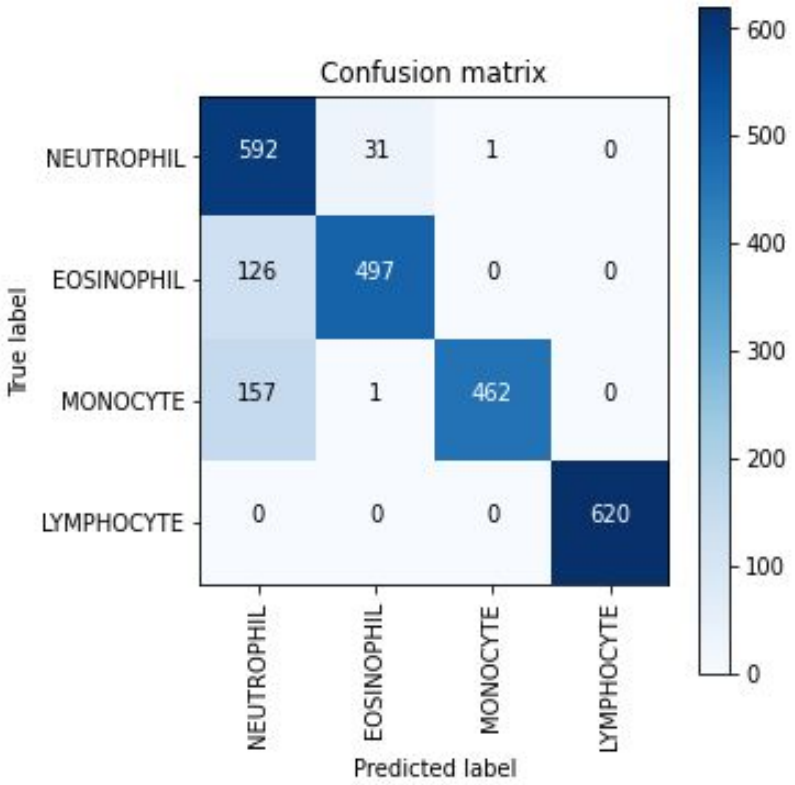


Performance

Test Accuracy = 87.29 %

f1-score: 0.876473181098623

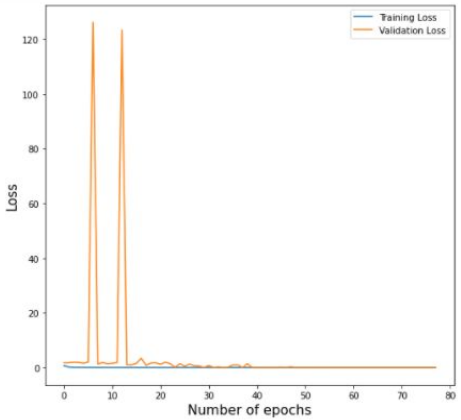
	precision	recall	f1-score	support
NEUTROPHIL	0.68	0.95	0.79	624
EOSINOPHIL	0.94	0.80	0.86	623
MONOCYTE	1.00	0.75	0.85	620
LYMPHOCYTE	1.00	1.00	1.00	620
accuracy			0.87	2487
macro avg	0.90	0.87	0.88	2487
weighted avg	0.90	0.87	0.88	2487

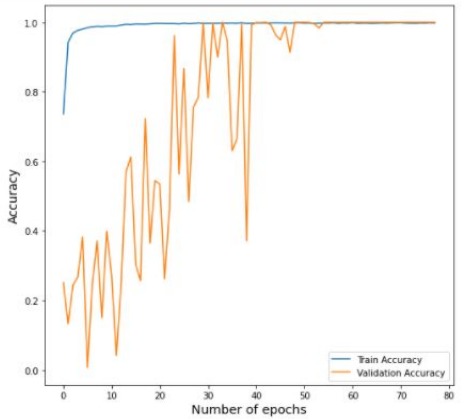


EfficientNet

- Test Accuracy: 88.31%
  (The curves seemed to be overfitting)

	precision	recall	f1-score	support
Category 1	0.70	0.96	0.81	624
Category 2	0.95	0.84	0.89	600
Category 3	1.00	0.73	0.84	620
Category 4	1.00	1.00	1.00	620
accuracy			0.88	2464
macro avg	0.91	0.88	0.89	2464
weighted avg	0.91	0.88	0.89	2464

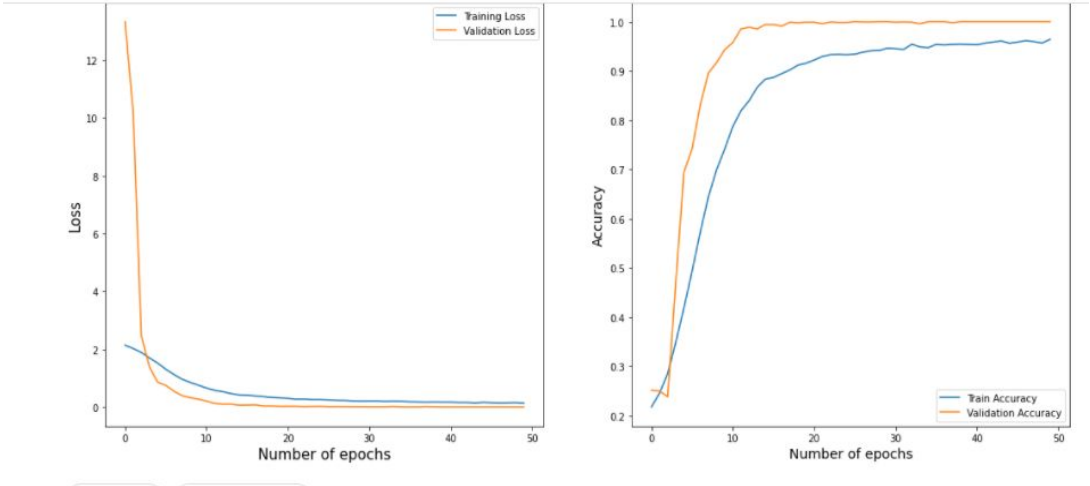




EfficientNet

- After trying to smooth the curve: The curves looked like underfitting

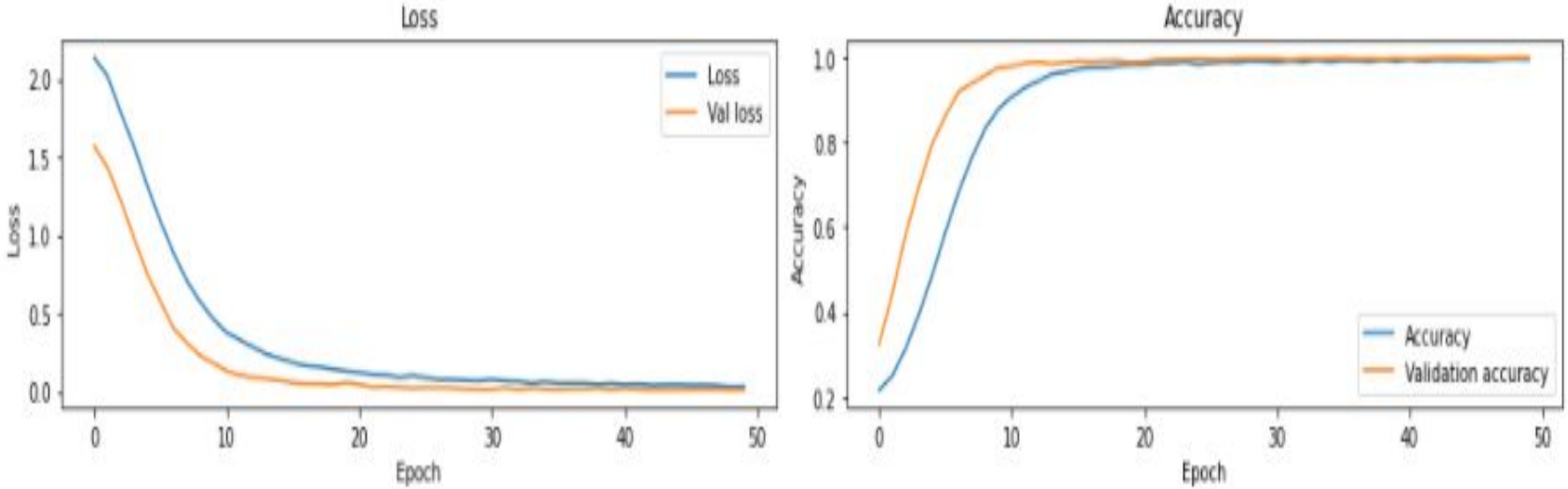
  (From google, we found that as EfficientNet has 217 layers so it needs too much data to be trained, but our data seemed not to be enough to use this model. So we discarded it.)



InceptionV3

Test Accuracy: 82.75 %

macro-f1: 0.8333315851211085



Challenges

- Shortage of ram
  - Used gradient accumulator
  - Used the saved best model
- Overfitting:
  - Used pretrained model
  - Froze the layers of pretrained model except the batch normalization layers
  - Added dropout and L2 regularization
  - Used Adam optimizer

Thank You!