Blood Cell Type Prediction

Deep Learning for Automated Blood Cell Classification

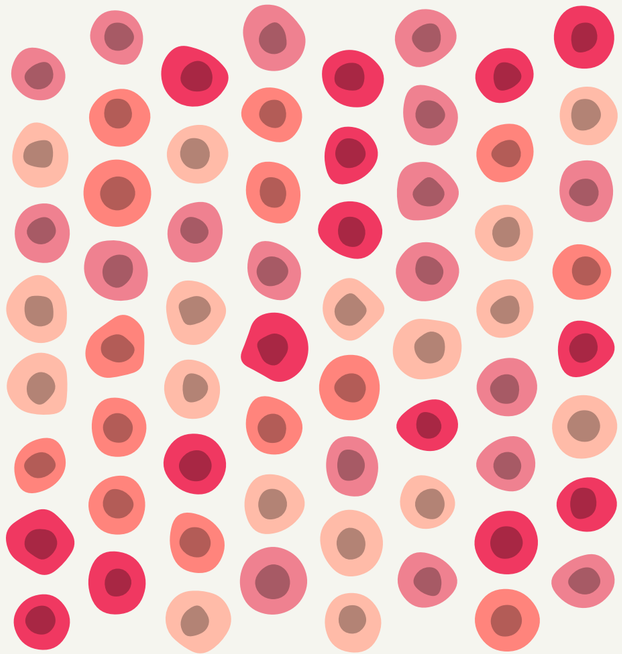
Machine Learning for Human Data 2024/25

Enxhi Nushi & Rana Islek

2024/25

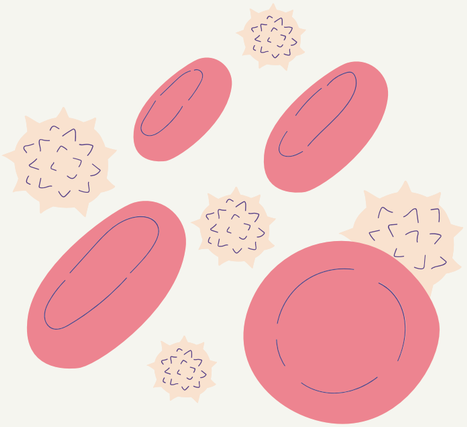
Outline

Part 1 - Introduction: Objectives & Dataset Overview
Part 2 - Model Architectures: Baseline CNN, ResNet, Inception, VGG
Part 3 - Experimentation & Findings: Classification Reports & Learning of Models
Part 4 - Model Comparison: Performance Evaluation & Confusion Matrix Analysis
Part 5 - Conclusion: Key Takeaways

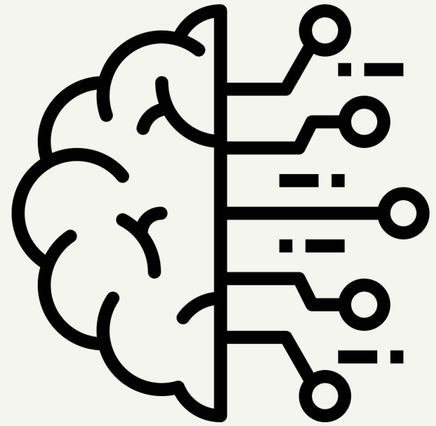


Introduction

Part 01



Introduction



Background & Motivation

- Importance of blood cell classification in hematological diagnosis
- Limitations of manual microscopy: time-consuming & expertise-dependent
- Role of deep learning in medical image analysis

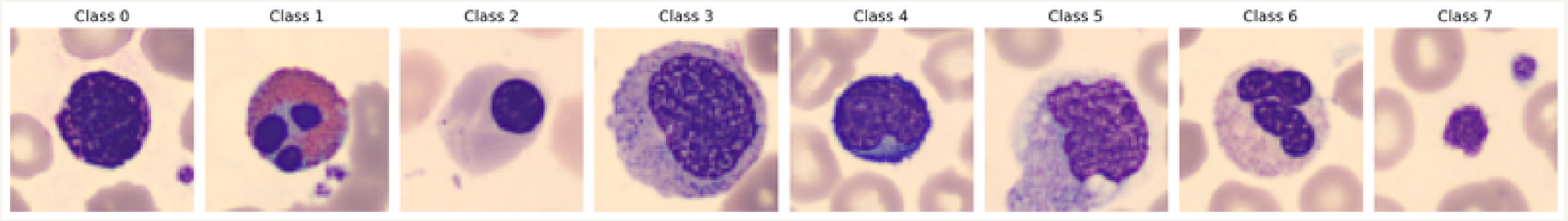
Research Objectives

- Compare multiple deep learning architectures for blood cell classification
- Evaluate performance in terms of accuracy, efficiency, and computational cost

Dataset: BloodMNIST

Dataset Overview

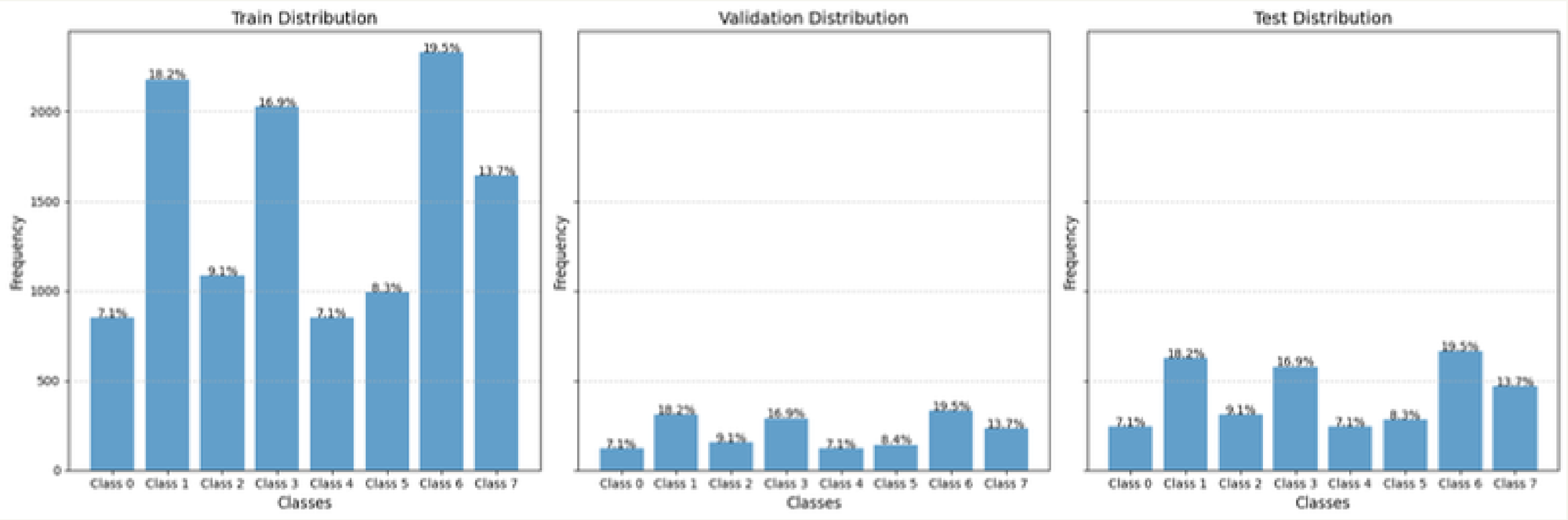
- **Source**: Part of the MedMNIST collection
- **Size**: 17,092 labeled images of 8 blood cell types
- **Image Format**: 64×64 RGB
- **Cell Types**: Basophils, Eosinophils, Erythroblasts, Lymphocytes, Monocytes, Neutrophils, Immature Granulocytes, Platelets



Dataset: BloodMNIST

Preprocessing Techniques

- Normalization (scaling pixel values to [0,1])
- Already splitted into Train (11,959), Validation (1,712), Test (3,421) -> 70-10-20

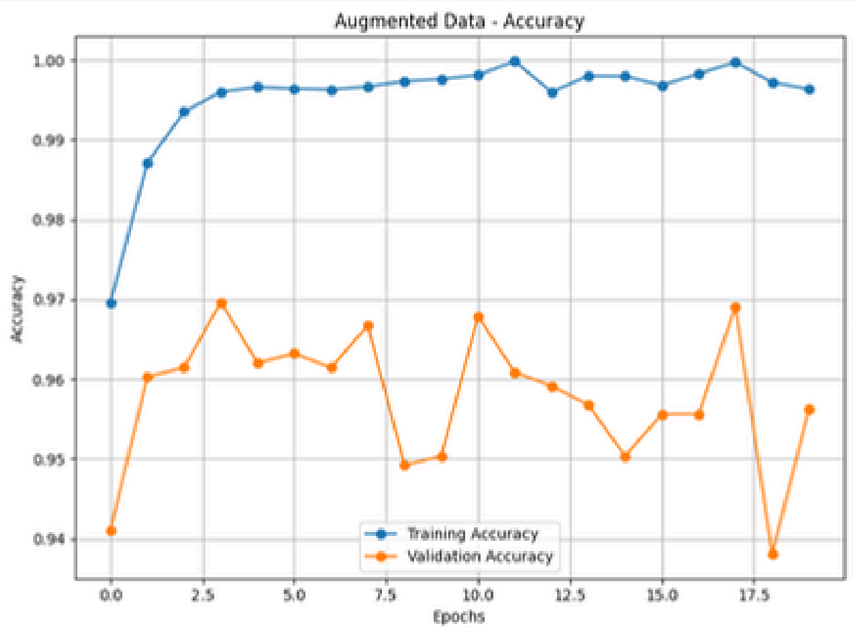


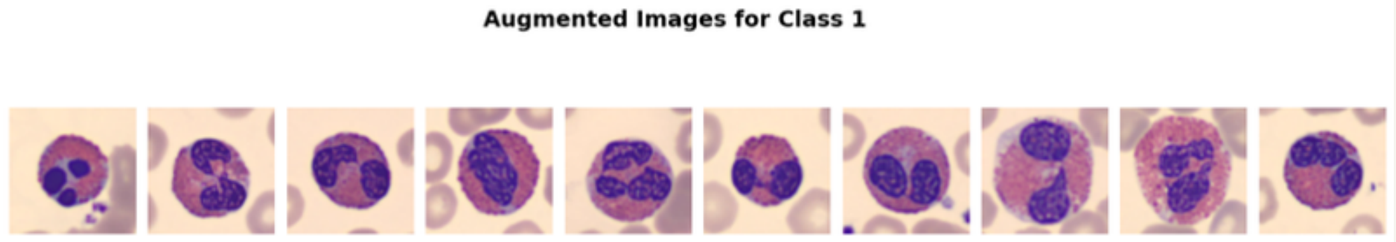
Dataset: BloodMNIST

Preprocessing Techniques

- Data Augmentation experiments (rotation, flipping) and its impact on accuracy:

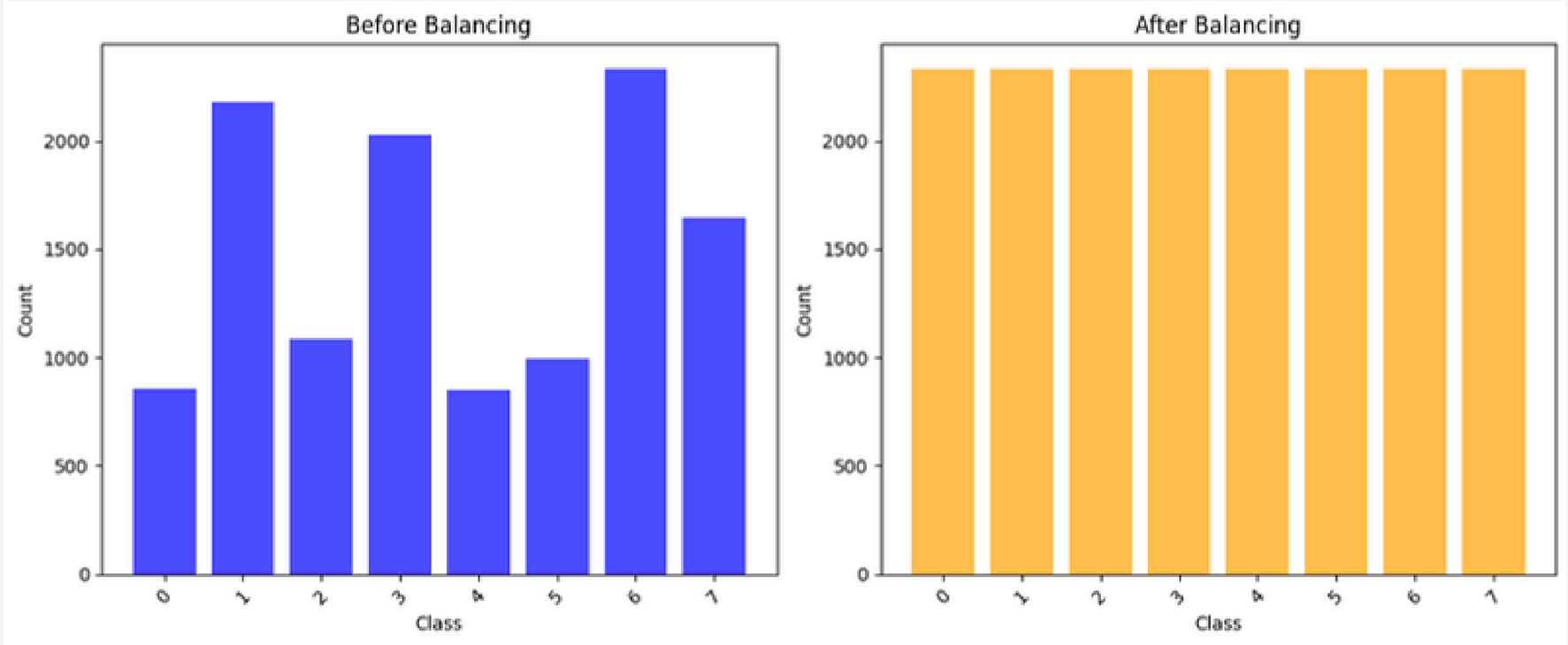
WARNING: We decided not to train our models with augmented data to ensure that they learn from authentic, unaltered representations of blood cells.





Dataset: BloodMNIST

Preprocessing Techniques



Research Questions

What we want to experiment?

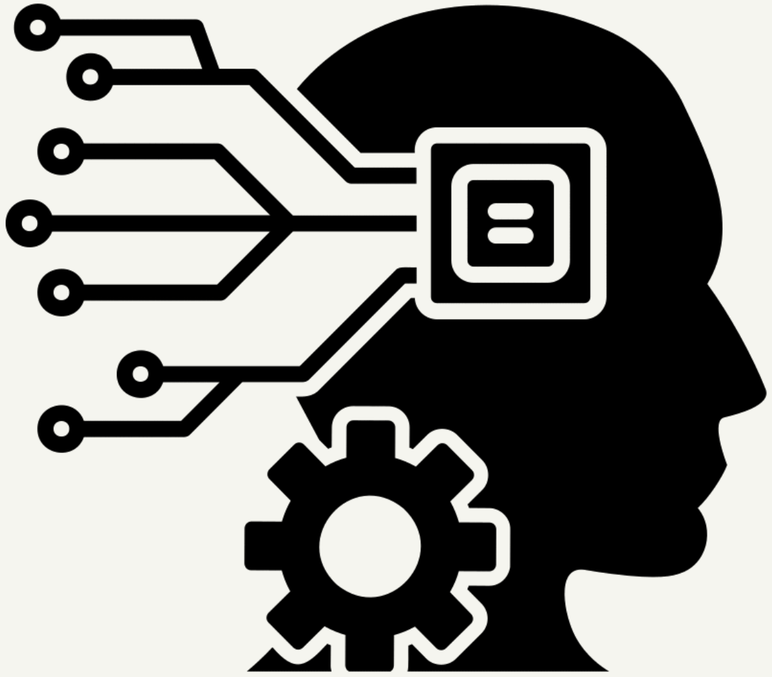
Q1: Can a **lightweight model** with fewer parameters achieve competitive accuracy?

Q2: Can **attention mechanisms** (SE, CBAM) enhance performance?

Q3: How do **complex architectures** (ResNet, Inception-v3, VGG-16) compare in terms of accuracy vs computational efficiency?

Model Architectures

Part 02



Model Architectures Explored

Baseline Models

Simple CNN & Variations:

2-layer

3-layer

4-layer

Deep Architectures

ResNet

Inception-v3

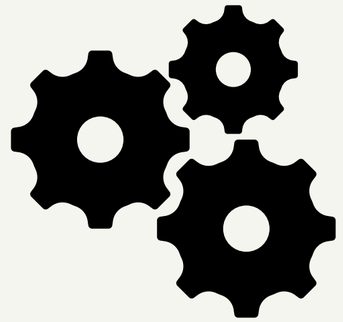
VGG-16

Attention Mechanisms

Squeeze-and-Excitation **(SE)** Networks

Convolutional Block Attention Module **(CBAM)**

Model Optimization Techniques



Regularization Methods

Dropout (0.3-0.5):
- Prevents overfitting

Batch Normalization:
- Stabilizes gradient updates

L2 Weight Regularization:
(λ = 0.0001 - 0.01)
- Reduces large weight values

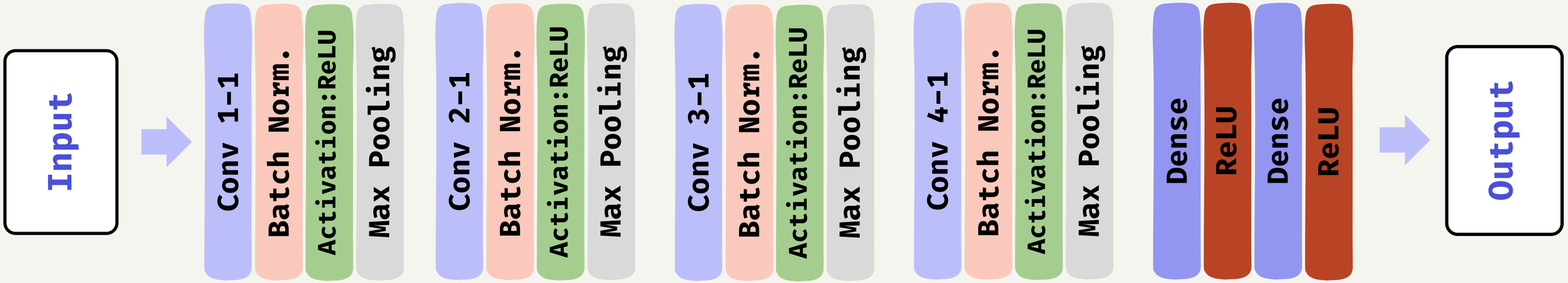
Training Strategy

Adam & SGD optimizers
Learning rate scheduling
Early stopping for overfitting prevention

Experimentation & Findings Part 03

1.CNN

CNN Model Architecture



📌 **Convolutional Layers:** Extract spatial features using small filters. Detects edges, textures, and patterns progressively.
📌 **Batch Normalization:** Stabilizes activations and speeds up training. Helps reduce internal covariate shift.
📌 **ReLU Activation:** Introduces non-linearity for better learning. Prevents vanishing gradient issues.
📌 **Max Pooling:** Downsamples feature maps to reduce computation. Retains the most important features.
📌 **Fully Connected Layers (Dense Layers):** Interprets extracted features for classification. Uses ReLU activation before final prediction.

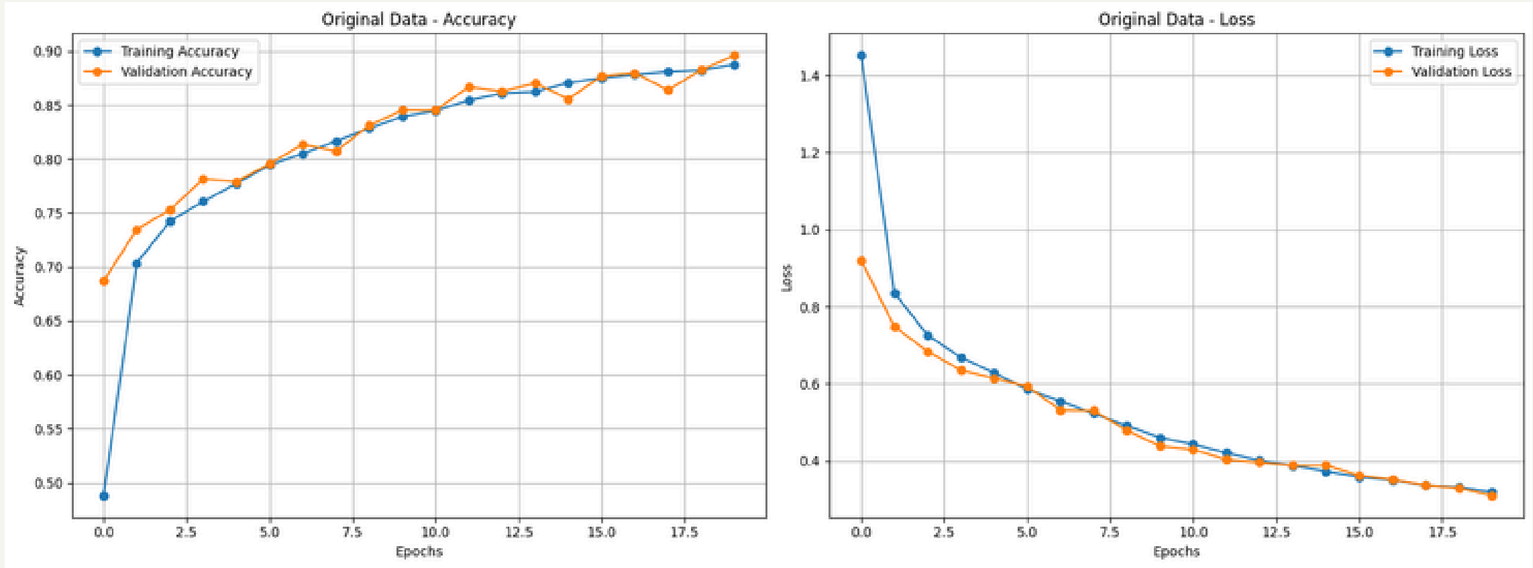
1.1 Baseline CNN Evolution - Experimenting Layers

- 2-layer, 3-layer, 4-layer models compared
- Best accuracy: 4-layer CNN (88.78%)
- Total params: 130792 (510.91 KB), Trainable params: 130792 (510.91 KB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.83	0.82	0.83	244
class1	0.98	0.97	0.97	624
class2	0.90	0.87	0.89	311
class3	0.76	0.73	0.75	579
class4	0.93	0.81	0.86	243
class5	0.69	0.78	0.74	284
class6	0.93	0.98	0.95	666
class7	0.99	0.99	0.99	470

Test Accuracy: 88.78%
Weighted F1 Score: 0.8877
Weighted Recall: 0.8878
Weighted Precision: 0.8891



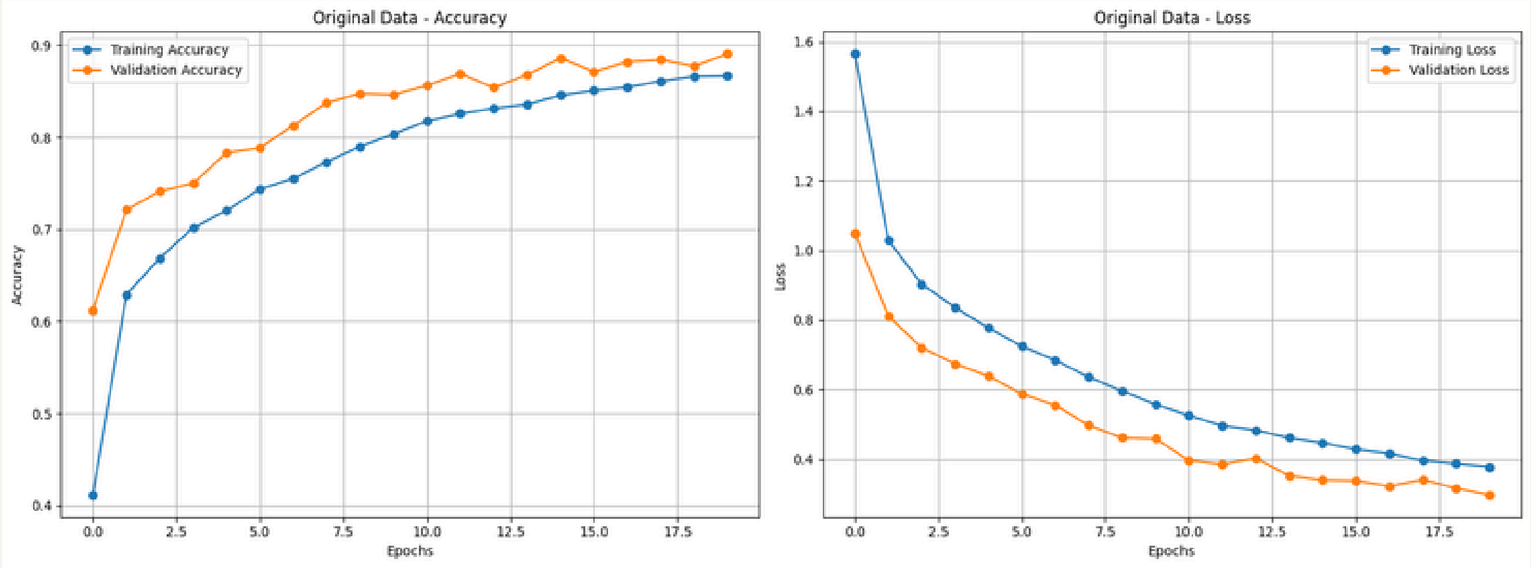
1.2 CNN Evolution - Experimenting Dropout Layers

- Dropout Regularization (0.4, 0.5) after the dense layer
- Best accuracy: with 0.4 dropout rate (91.9%)
- Total params: 549032 (2.09 MB), Trainable params: 549032 (2.09 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.93	0.90	0.91	244
class1	0.97	0.99	0.98	624
class2	0.92	0.83	0.87	311
class3	0.79	0.89	0.84	579
class4	0.93	0.92	0.92	243
class5	0.91	0.71	0.80	284
class6	0.95	0.97	0.96	666
class7	0.98	0.99	0.98	470

Test Accuracy: 91.90%
Weighted F1 Score: 0.9185
Weighted Recall: 0.9190
Weighted Precision: 0.9214



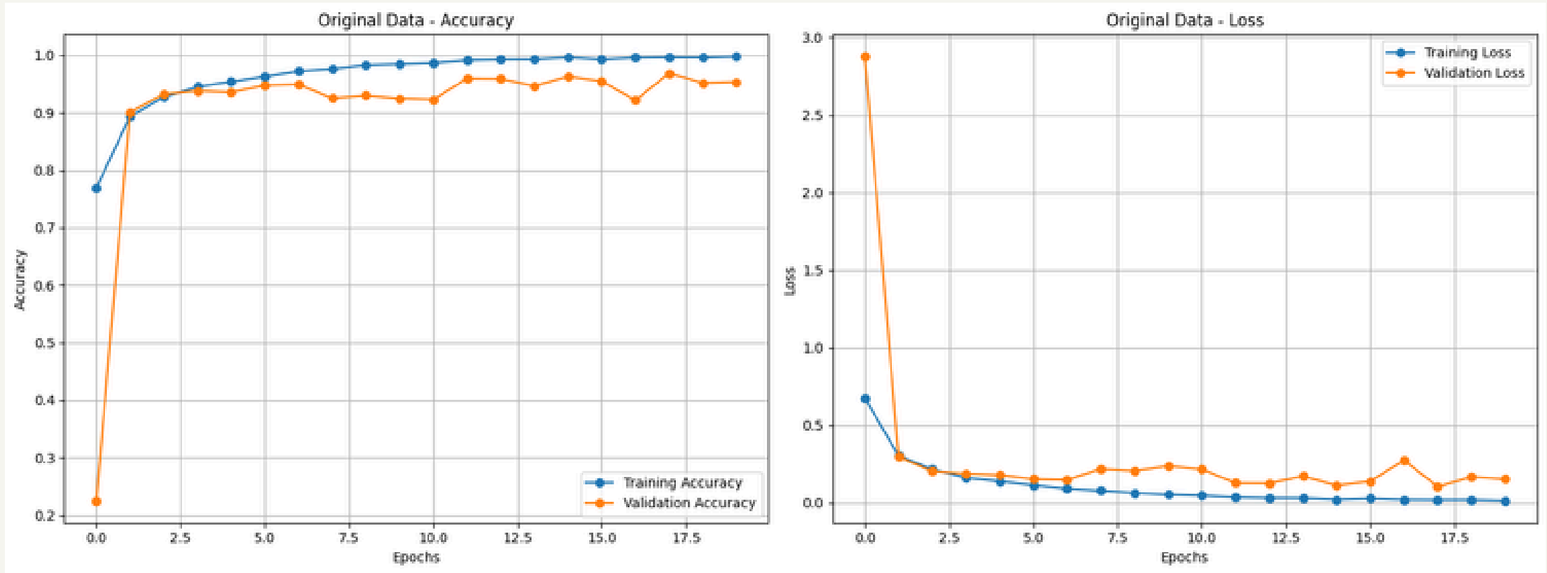
1.3 CNN Evolution - Experimenting Batch Normalization

- Batch Normalization (with dropout (0.4))
- Best accuracy: (95.59%)
- Total params: 626024 (2.39 MB), Trainable params: 625032 (2.38 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.97	0.96	0.97	244
class1	0.99	1.00	1.00	624
class2	0.97	0.97	0.97	311
class3	0.97	0.82	0.88	579
class4	0.97	0.98	0.97	243
class5	0.89	0.95	0.92	284
class6	0.90	0.99	0.95	666
class7	1.00	1.00	1.00	470

Test Accuracy: 95.59%
Weighted F1 Score: 0.9551
Weighted Recall: 0.9559
Weighted Precision: 0.9575



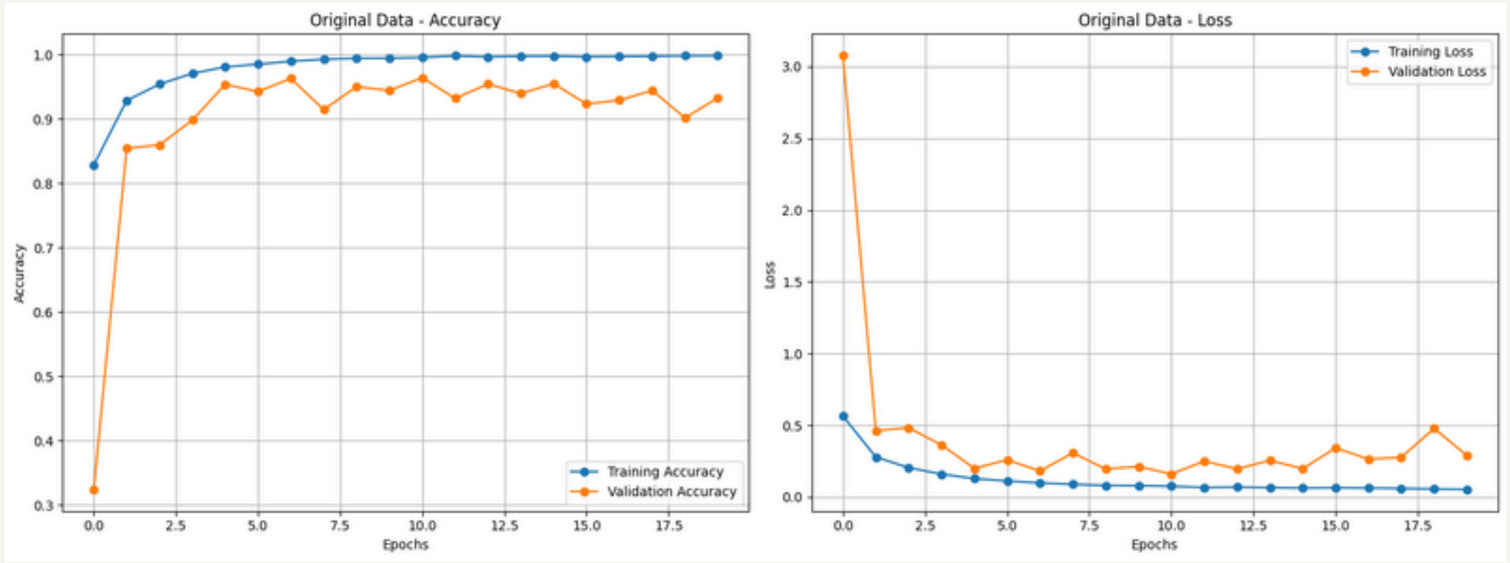
1.4 CNN Evolution - Experimenting L2 Weight Regularization

- L2 Weight Regularization (with rates (0.01, 0.001, 0.0001))
- Best accuracy: (91.11%) with 0.0001 -> not improved by its own
- Total params: 626024 (2.39 MB), Trainable params: 625032 (2.38 MB) -> same

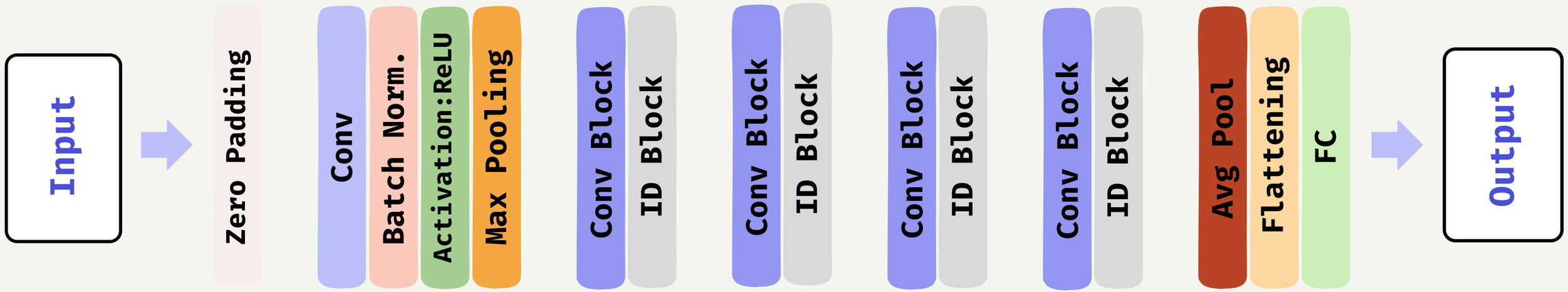
Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.98	0.55	0.70	244
class1	1.00	0.97	0.98	624
class2	0.99	0.91	0.95	311
class3	0.69	0.97	0.81	579
class4	0.92	0.98	0.95	243
class5	0.95	0.83	0.89	284
class6	0.99	0.88	0.93	666
class7	0.99	1.00	0.99	470

Test Accuracy: 91.11%
Weighted F1 Score: 0.9116
Weighted Recall: 0.9111
Weighted Precision: 0.9315



2. Comparison of Deep Architectures - ResNet



📌 **Zero Padding:** Adds extra zero-valued pixels around an image before applying convolution. Preserves edge features that might be lost.
📌 **Residual Connections (Skip Connections):** Bypasses layers to allow direct gradient flow. Prevents vanishing gradients in deep networks.
📌 **Convolutional Blocks (Conv Blocks):** Standard convolutional layers with batch normalization & ReLU. Extracts hierarchical image features.
📌 **Identity Blocks (ID Blocks):** Preserves information by skipping some layers. Helps deeper networks learn without degradation.
📌 **Batch Normalization:** Speeds up training and stabilizes activations. Improves generalization across datasets.
📌 **Global Average Pooling & Fully Connected Layer:** Replaces dense layers to reduce overfitting. Final fully connected layer for classification.
📌 **Flattening:** Converts a multi-dimensional feature map into a 1D vector. Prepares the extracted features for fully connected (dense) layers.

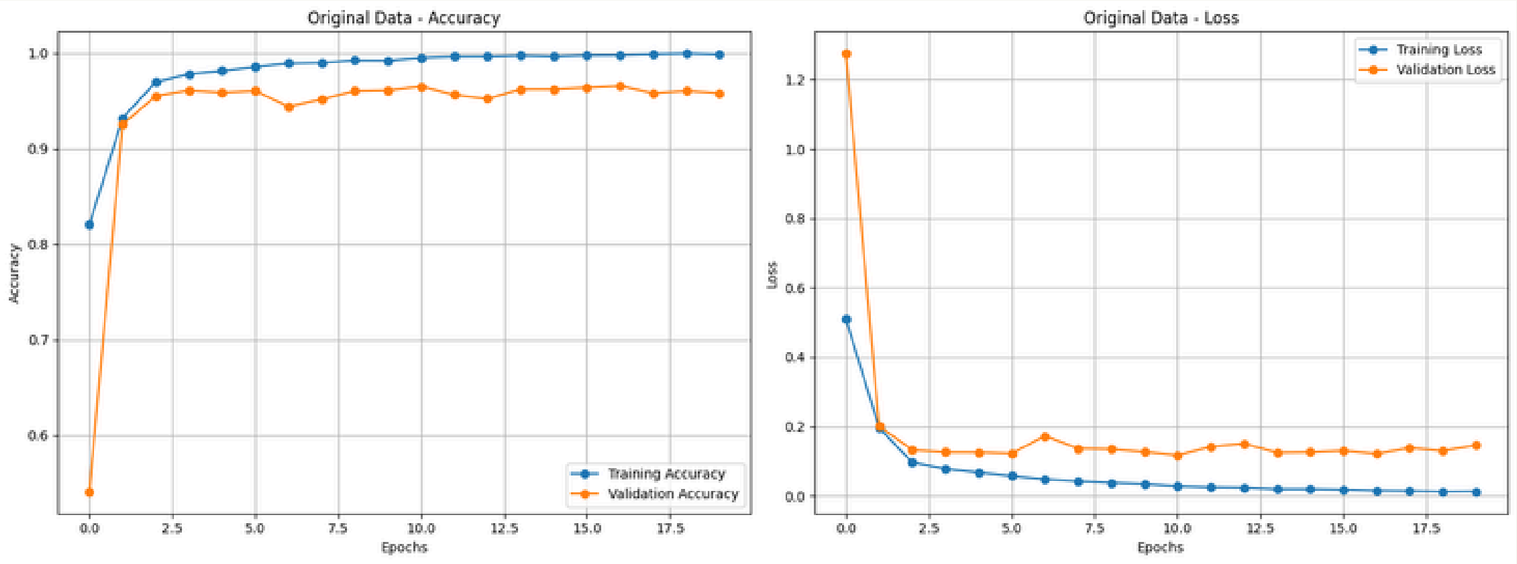
2.1 Comparison of Deep Architectures - ResNet

- Best performance: ResNet (95.67%)
- SGD as optimizer with LR scheduler
- Total params: 11195016 (42.71 MB), Trainable params: 11185416 (42.67 MB) -> **big !!!**

Detailed Classification Report:

	precision	recall	f1-score	support
class0	1.00	0.92	0.96	244
class1	0.99	0.99	0.99	624
class2	0.96	0.95	0.96	311
class3	0.92	0.89	0.91	579
class4	0.94	0.99	0.97	243
class5	0.84	0.94	0.89	284
class6	0.97	0.96	0.97	666
class7	1.00	1.00	1.00	470

Test Accuracy: 95.67%
Weighted F1 Score: 0.9570
Weighted Recall: 0.9567
Weighted Precision: 0.9581



2.2 Comparison of Deep Architectures - Inception-v3

Unlike traditional CNNs with sequential convolutional layers, **Inception-v3 uses parallel convolutional paths** with different filter sizes within each block. This multi-branch approach allows the model to **capture both fine-grained and high-level features simultaneously**.

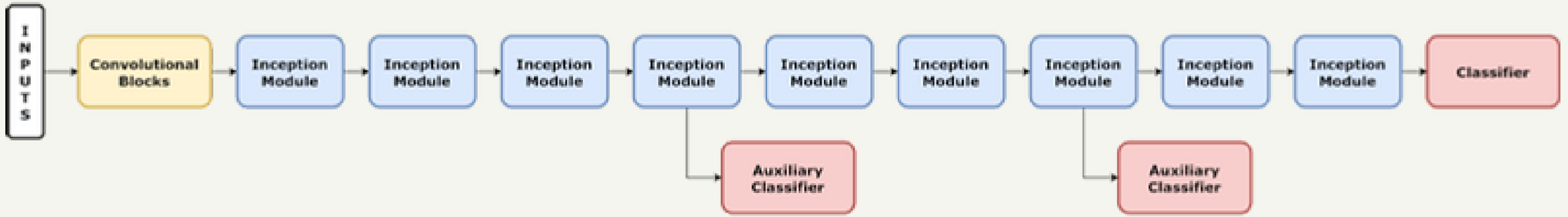
📌**1. Stem Block**
- Initial convolutional layers for basic feature extraction.
- Uses Batch Norm + ReLU for stable training.

📌**2. Inception Modules**
- Parallel convolutions with different filter sizes (1×1, 3×3, 5×5).
- Captures multi-scale features efficiently.
- Depthwise separable convolutions reduce computation.

📌**3. Reduction Blocks**
- Downsampling layers with stride-2 convolutions & pooling.
- Preserves features while lowering computation cost.

📌**4. Auxiliary Classifiers**
- Intermediate outputs used as side classifiers.
- Helps with gradient flow & prevents vanishing gradients.

📌**5. Global Average Pooling & Fully Connected Layer**
- Global Average Pooling minimizes overfitting.
- Final fully connected layer with softmax for classification of eight cell types.



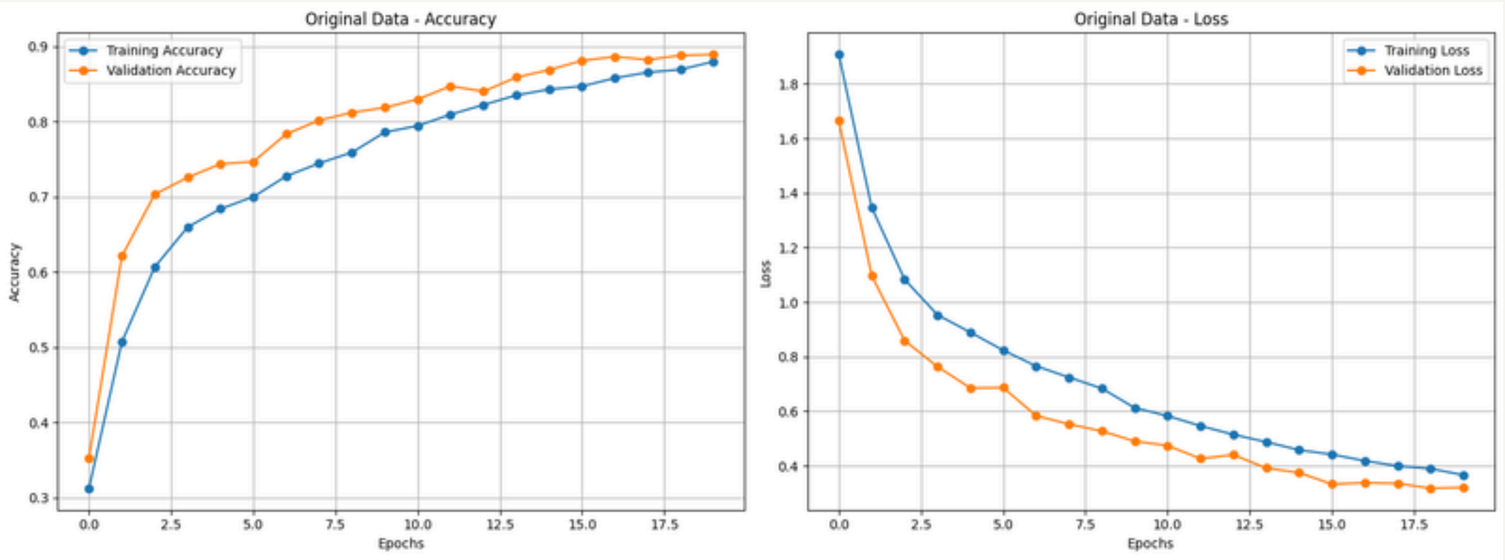
2.2 Comparison of Deep Architectures - Inception-v3

- Best performance: Inception-v3 underperformed (88.02%)
- Total params: 10313800 (39.34 MB), Trainable params: 10288520 (39.25 MB) **-> big !!!**

Detailed Classification Report:

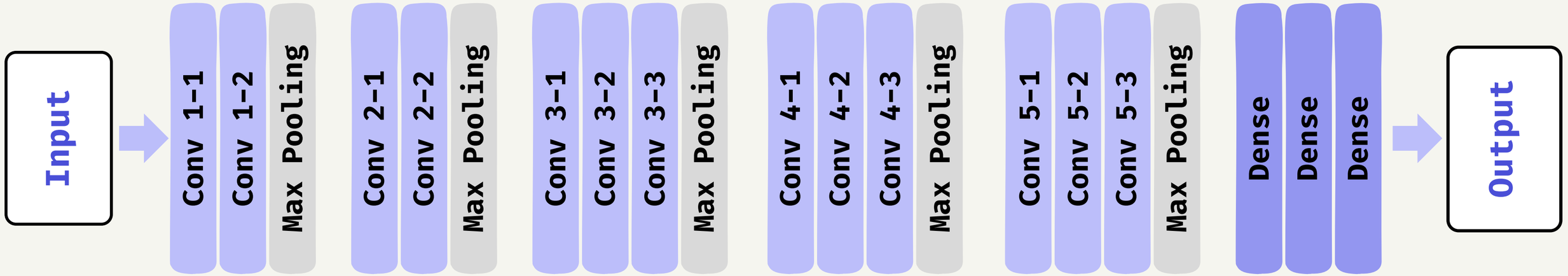
	precision	recall	f1-score	support
class0	0.83	0.78	0.80	244
class1	0.92	0.99	0.96	624
class2	0.99	0.86	0.92	311
class3	0.69	0.81	0.74	579
class4	0.96	0.68	0.80	243
class5	0.83	0.68	0.75	284
class6	0.92	0.96	0.94	666
class7	1.00	1.00	1.00	470

Test Accuracy: 88.02%
Weighted F1 Score: 0.8796
Weighted Recall: 0.8802
Weighted Precision: 0.8870



2.3 Comparison of Deep Architectures - VGG16

VGG-16 (Visual Geometry Group) Model Architecture



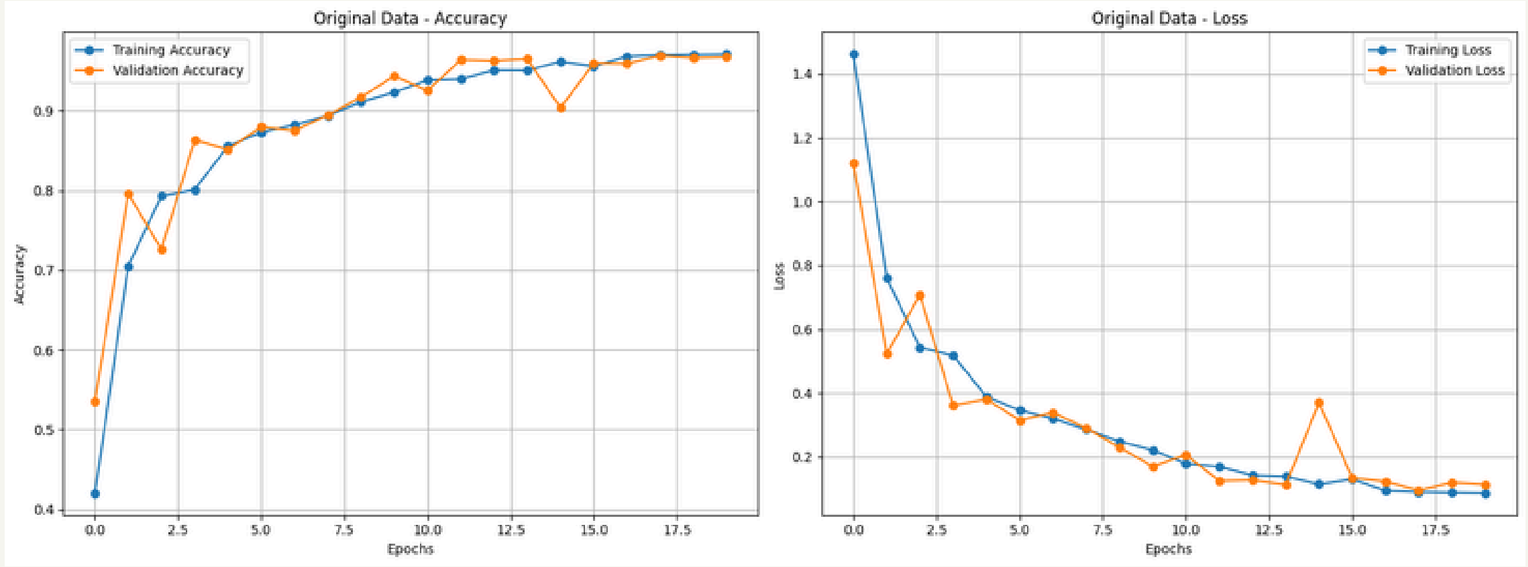
📌 **Deep Convolutional Architecture:** 16 layers deep (13 convolutional + 3 dense layers). Uses small 3×3 filters for better feature extraction.
📌 **Stacked Convolutional Layers:** Multiple convolutional layers per block enhance hierarchical learning. Followed by Max Pooling (2×2, stride 2) for downsampling.
📌 **Uniform Structure:** All convolutional layers use 3×3 filters & ReLU activation. Maintains simplicity while improving performance.
📌 **Fully Connected (Dense) Layers:** Three dense layers process extracted features for classification. Uses ReLU activation, followed by a softmax output layer.

2.3 Comparison of Deep Architectures - VGG16

- Dropout layer added after the dense layers
- Best performance: VGG-16 (96.43%)
- Total params: 5787560 (22.08 MB), Trainable params: 5787560 (22.08 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.93	0.98	0.95	244
class1	0.99	1.00	0.99	624
class2	0.99	0.92	0.95	311
class3	0.93	0.91	0.92	579
class4	0.96	0.99	0.97	243
class5	0.96	0.94	0.95	284
class6	0.95	0.98	0.97	666
class7	0.99	1.00	0.99	470



Test Accuracy: 96.43%
Weighted F1 Score: 0.9641
Weighted Recall: 0.9643
Weighted Precision: 0.9645

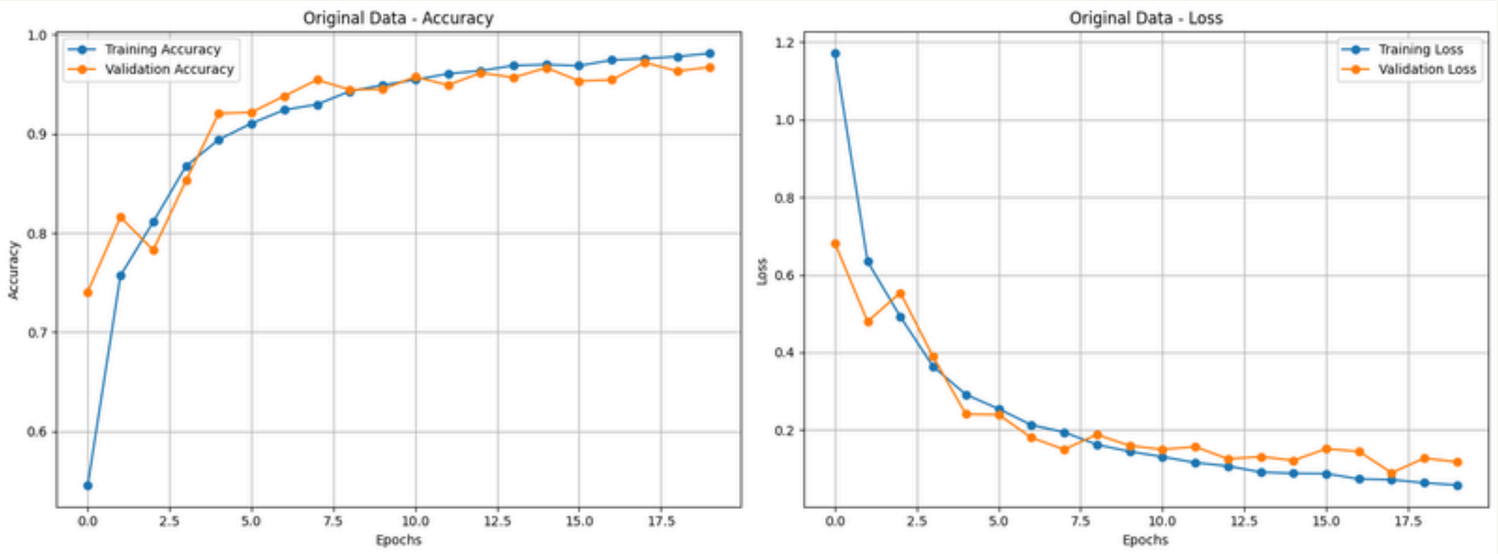
2.4 Comparison of Deep Architectures - VGG16

- Removing layers from 4th and 5th block
- Best performance: VGG-16 (96.20%)
- Total params: 4607400 (17.58 MB), Trainable params: 4607400 (17.58 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.96	0.97	0.97	244
class1	1.00	1.00	1.00	624
class2	0.98	0.89	0.93	311
class3	0.90	0.93	0.91	579
class4	0.96	0.98	0.97	243
class5	0.96	0.91	0.94	284
class6	0.96	0.98	0.97	666
class7	1.00	1.00	1.00	470

Test Accuracy: 96.20%
Weighted F1 Score: 0.9620
Weighted Recall: 0.9620
Weighted Precision: 0.9626

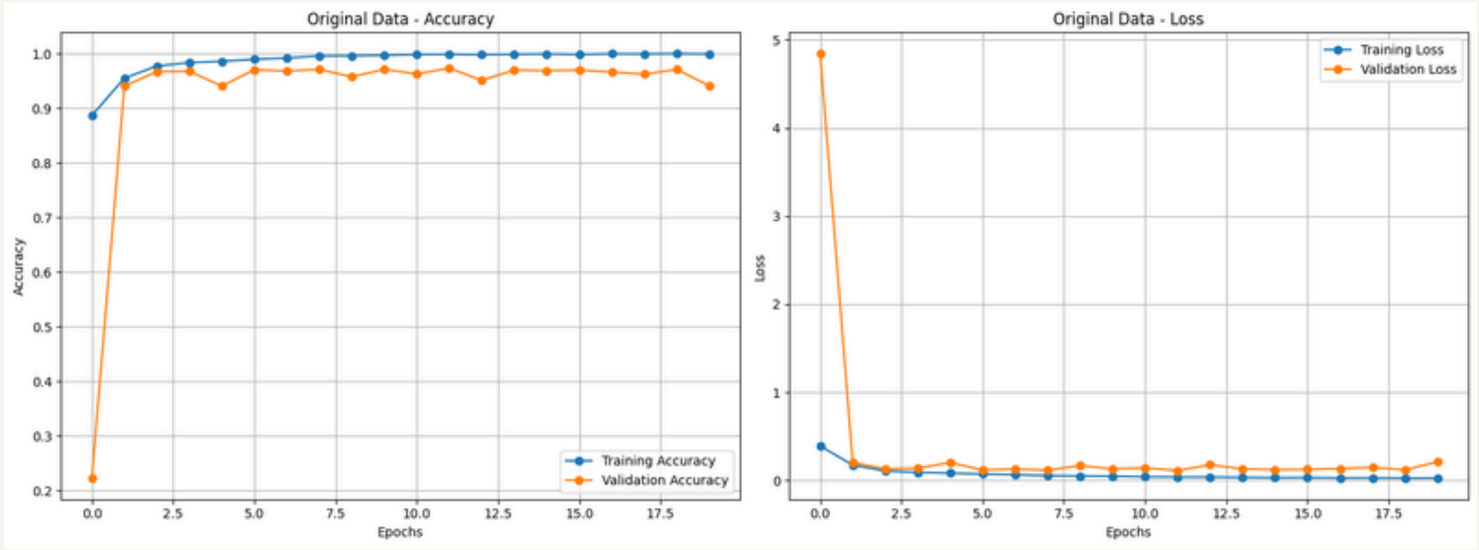


3.1 Impact of Attention Mechanisms: CNN

- Updated CNN with Batch Normalization, L2 Regularization and Attention (SE)
- SE improved CNN performance to 95.03% test accuracy
- Total params: 287807 (1.10 MB), Trainable params: 287455 (1.10 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.99	0.93	0.96	244
class1	0.99	1.00	0.99	624
class2	0.87	0.99	0.93	311
class3	0.94	0.85	0.89	579
class4	0.95	0.95	0.95	243
class5	0.87	0.94	0.90	284
class6	0.96	0.96	0.96	666
class7	0.99	1.00	1.00	470



Test Accuracy: 95.03%
Weighted F1 Score: 0.9501
Weighted Recall: 0.9503
Weighted Precision: 0.9516

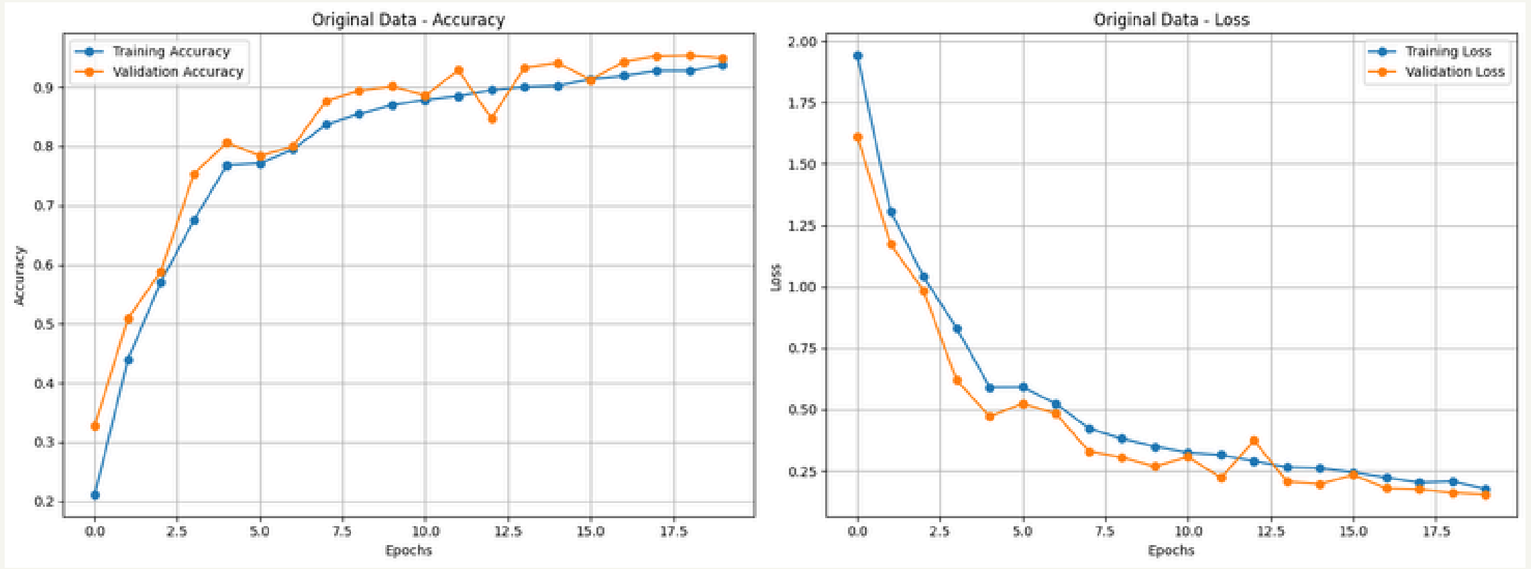
3.2 Impact of Attention Mechanisms: VGG16

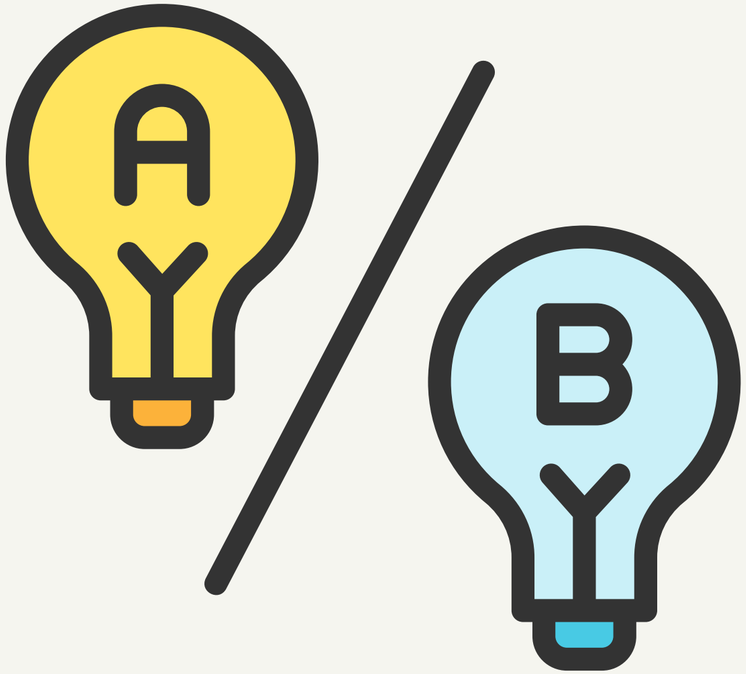
- CBAM improved VGG performance to 93.31% test accuracy
- Total params: 5800994 (22.13 MB), Trainable params: 5800994 (22.13 MB)

Detailed Classification Report:

	precision	recall	f1-score	support
class0	0.86	0.95	0.91	244
class1	1.00	0.98	0.99	624
class2	0.97	0.89	0.93	311
class3	0.80	0.93	0.86	579
class4	0.95	0.98	0.97	243
class5	0.95	0.73	0.82	284
class6	0.97	0.94	0.95	666
class7	1.00	0.99	0.99	470

Test Accuracy: 93.31%
Weighted F1 Score: 0.9333
Weighted Recall: 0.9331
Weighted Precision: 0.9387





Model Comparison

Part 04

Performance Evaluation

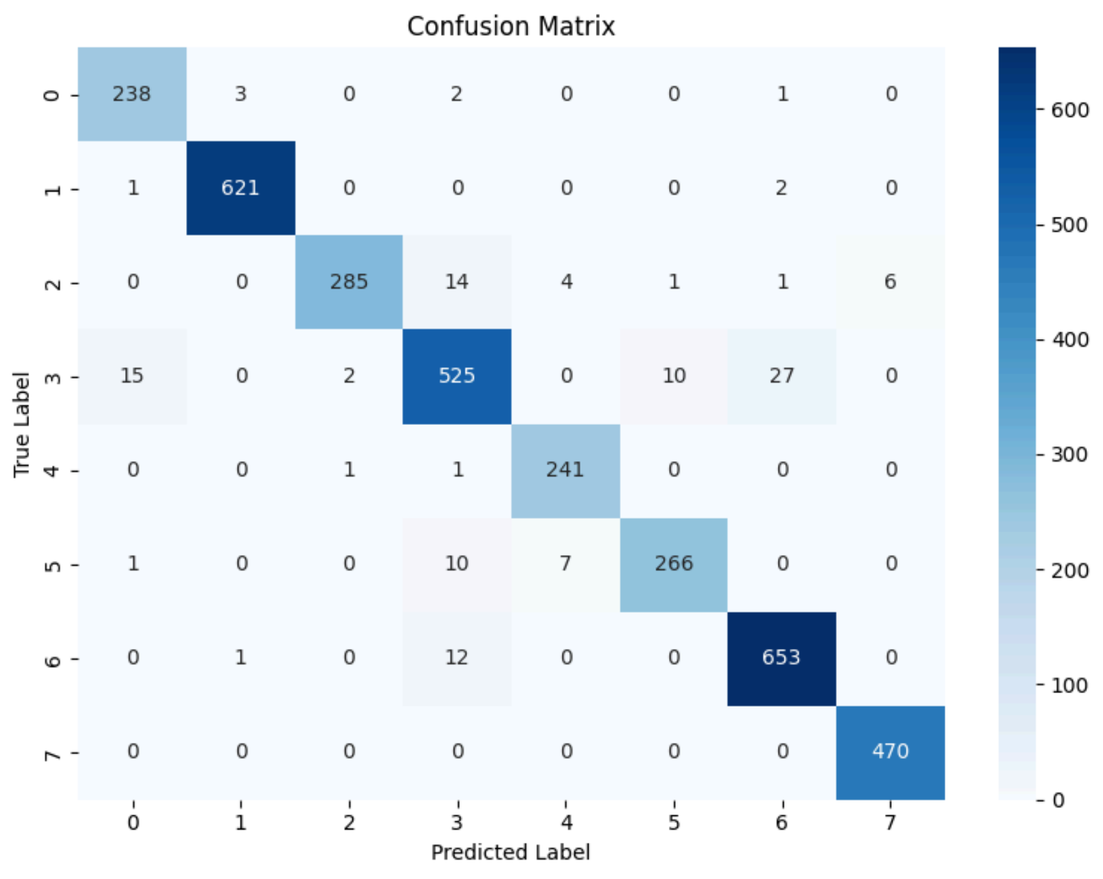
Comparison of Models Based on Accuracy

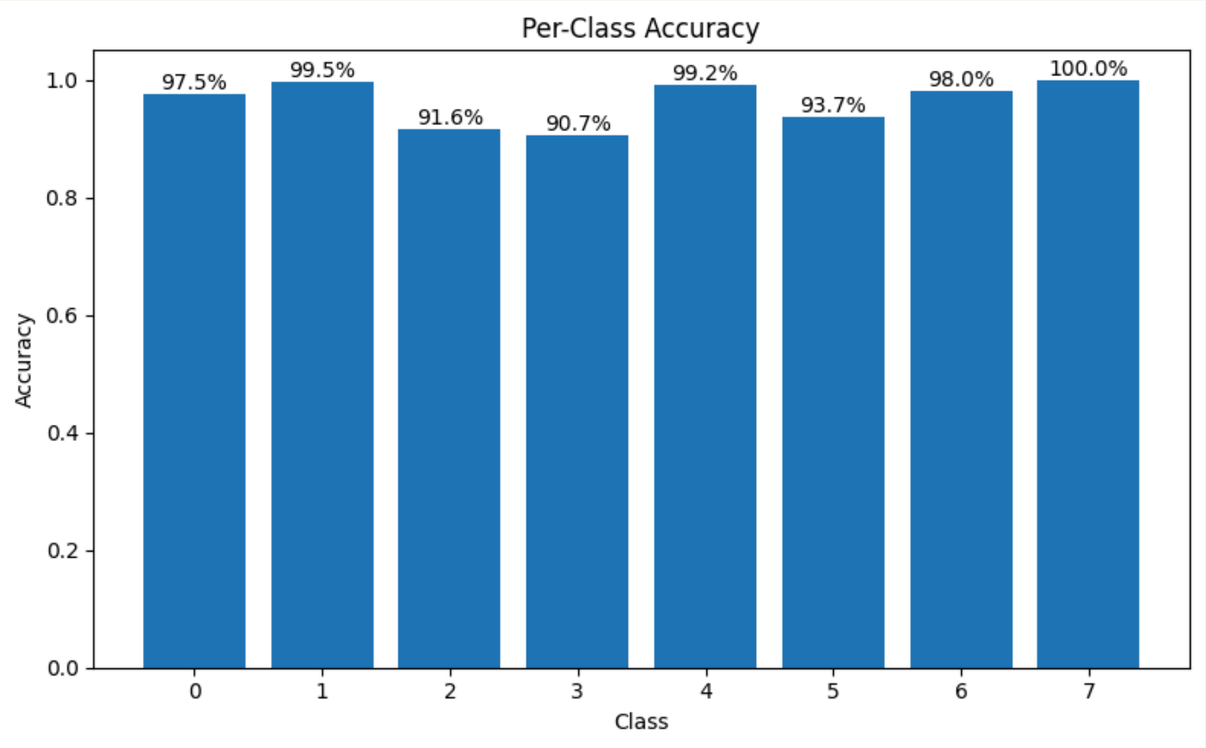
	Model Name	Accuracy (%)	F1-Score (%)	Execution Time (ms/step)	Memory Size (MB)
0	Basic CNN 3 Layers	88.07	88.07	6.05	0.67
1	Basic CNN 4 Layers	88.78	88.77	6.65	0.51
2	CNN 4 Layers with Batch Normalization	95.59	95.51	8.85	2.39
3	CNN with Attention L2	95.03	95.01	15.30	1.10
4	Restnet SGD	95.67	95.70	10.35	42.71
5	VGG-16	95.03	95.06	21.58	22.08
6	VGG-16 DL	96.43	96.41	21.75	22.08
7	VGG-16 Rm	96.20	96.20	20.69	17.58

Confusion Matrix Analysis

VGG-16 with Dropout (Best Model) Confusion Matrix: (0.96 test accuracy)

Shows high precision in all classes but some misclassification in similar blood cells

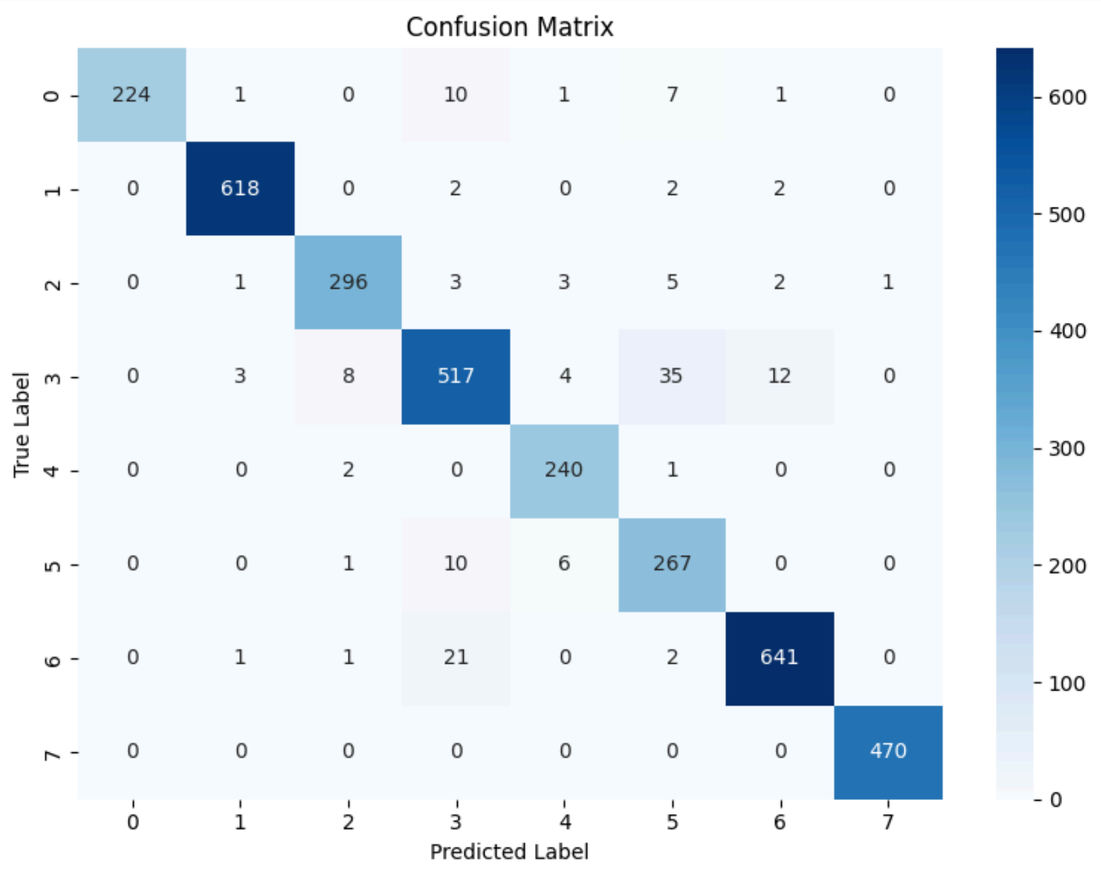


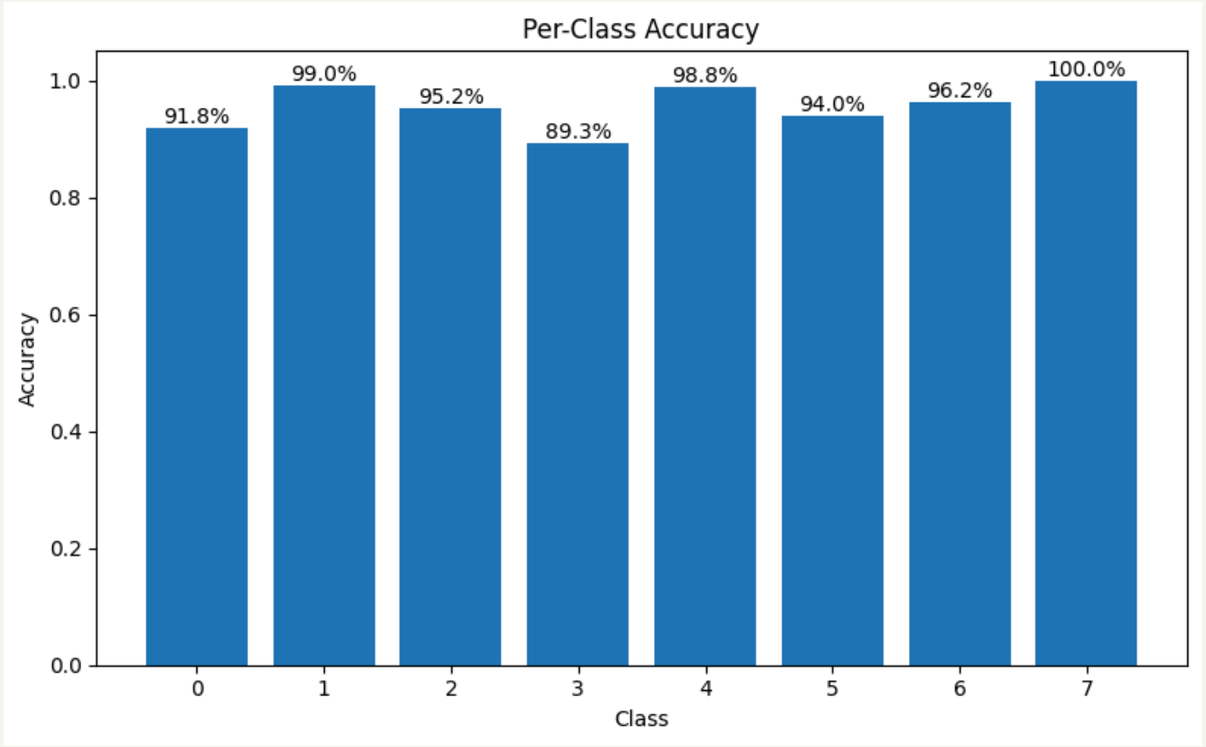


Confusion Matrix Analysis

ResNet (Best Model) Confusion Matrix: (0.96 test accuracy)

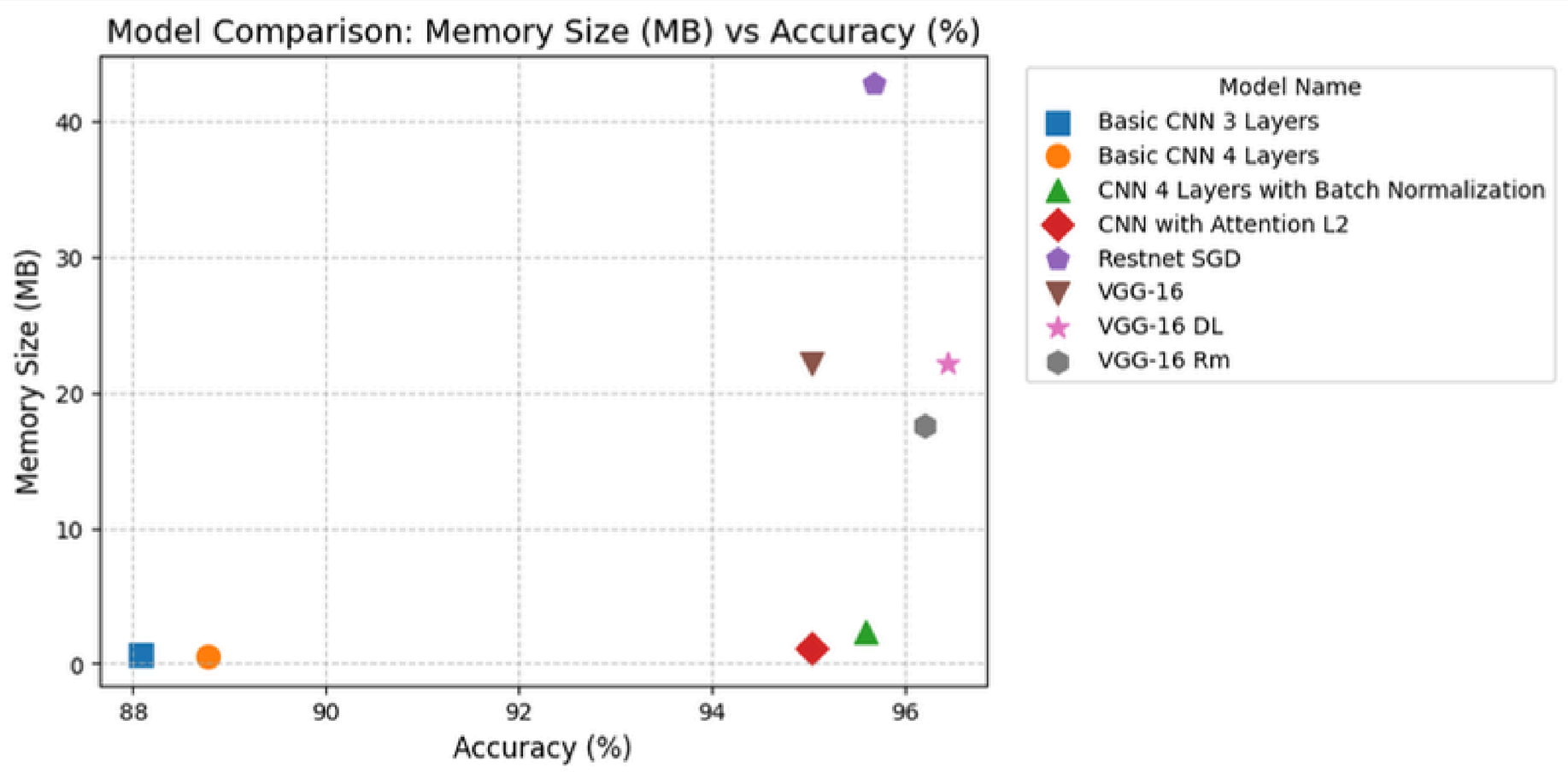
Competitive accuracy with slight misclassification in minority classes.





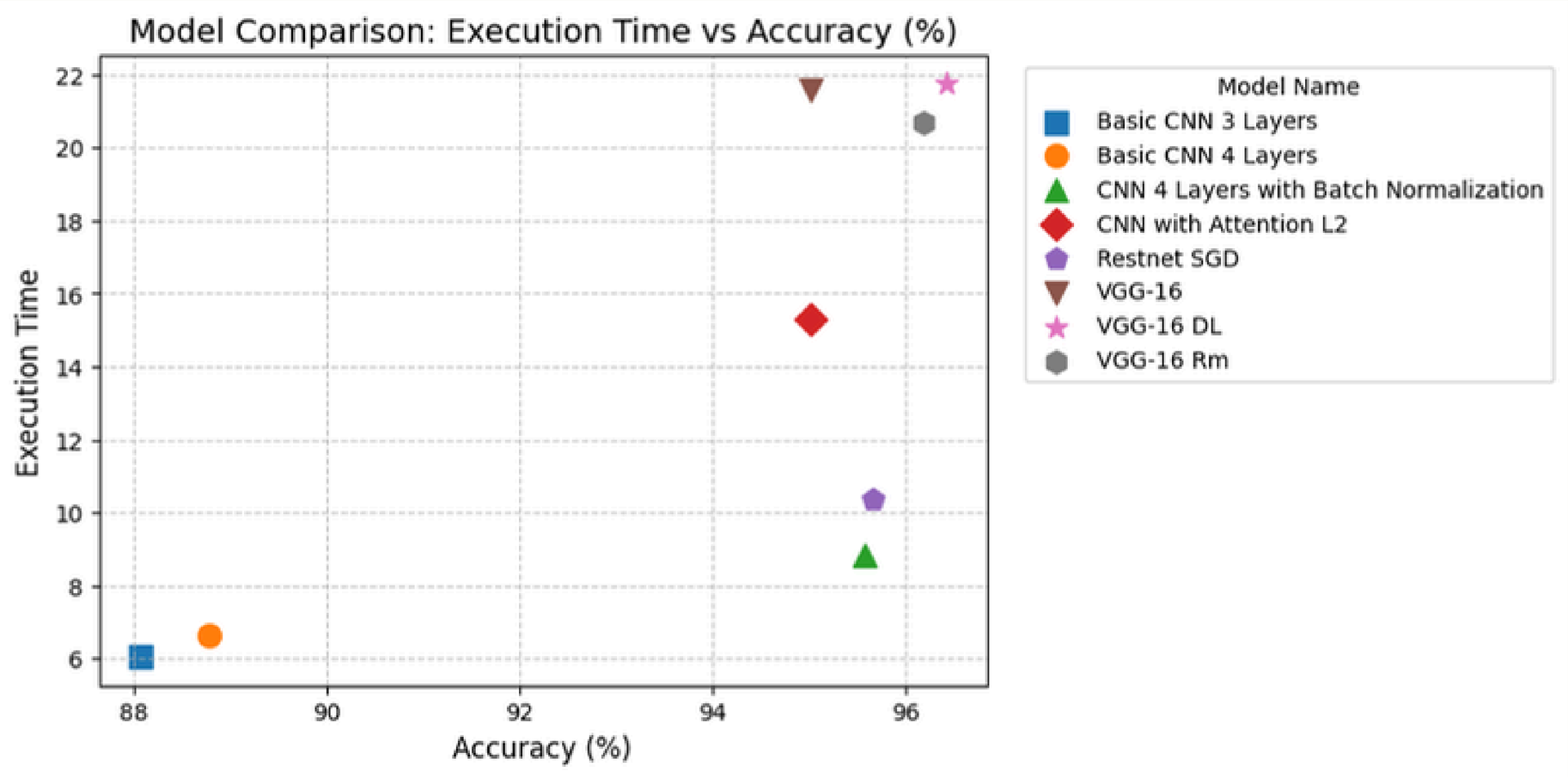
Computational Efficiency Considerations

Comparison of Accuracy & Memory Usage



Computational Efficiency Considerations

Comparison of Execution Time & Accuracy

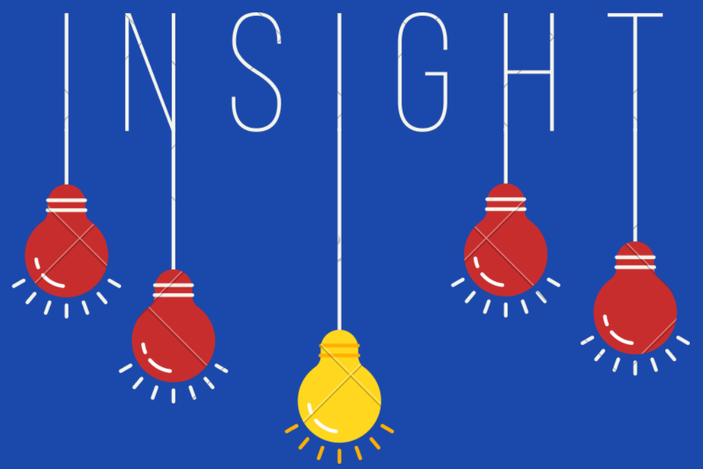


Conclusion

Part 05

Key Takeaways

Summary of Findings



Best Performing Model:

VGG-16 (96.43% ACCURACY) WITH DROPOUT LAYERS

Best Lightweight Model:

CNN WITH BATCH NORMALIZATION (95.59%)

Attention Mechanisms Boost Performance:

SE & CBAM IMPROVED ACCURACY

Computational Trade-offs Exist:

CHOOSING A MODEL DEPENDS ON ACCURACY VS EFFICIENCY

References

1. M. Habibzadeh, A. Krzyzak, and T. Fevens, “White Blood Cell Differ ential Counts Using Convolutional Neural Networks for Low Resolution Images,” in Artificial Intelligence and Soft Computing, (Springer).
2. A. Shahin, Y. Guo, K. Amin, and A. A. Sharawi, “White blood cells iden tification system based on convolutional deep neural learning networks,” Computer Methods and Programs in Biomedicine, vol. 168, pp. 69–80, Jan. 2019.
3. A. Acevedo, S. Alf´ erez, A. Merino, L. Puigv´ ı, and J. Rodellar, “Recogni tion of peripheral blood cell images using convolutional neural networks,” Data Brief, vol. 30, Aug. 2019.
4. A. S. B. Reddy and D. S. Juliet, “Transfer Learning with ResNet 50 for Malaria Cell-Image Classification,” International Conference on Communication and Signal Processing, Apr. 2019.
5. R. Liu, W. Dai, T. Wu, M. Wang, S. Wan, and J. Liu, “AIMIC: Deep Learning for Microscopic Image Classification,” Data Brief, vol. 226, Nov. 2022.