Technical Report for Assignment 2:

Building a single CNN network with multiclass classifier heads for 3 different classification tasks on the same data

Colab Notebook - DL_Assignment2.ipynb

Priyansh Saxena (B22EE075)

The report provides an in-depth analysis of the code, focusing on the **Main Assignment** and **Severity-Aware Bonus Implementation**.

The report is structured to cover the following aspects:

1. **Model Architecture and Parameters**
2. **Training Dynamics**
3. **Performance Metrics**
4. **Severity-Aware Implementation**
5. **Comparative Analysis**
6. **Critical Observations**
7. **Recommendations for Improvement**

---

1. Model Architecture and Parameters

1.1 Main Assignment Model

The **MultiHeadCNN** model is a custom convolutional neural network (CNN) designed for hierarchical classification on the CIFAR-100 dataset. It consists of:

- **Feature Extractor**:
  - Three convolutional blocks, each with two convolutional layers, batch normalization, ReLU activation, and max pooling.
  - Adaptive average pooling reduces spatial dimensions to 4x4.
- **Shared Fully Connected Layers**:
  - Two fully connected layers with dropout (0.5) for regularization.
- **Classification Heads**:
  - Three separate heads for class (100), superclass (20), and group (9) predictions.

Parameter Count:

- **Total Parameters**: 5,933,505
- **Trainable Parameters**: 5,933,505

1.2 Severity-Aware Model

The **SeverityAwareCNN** model is a simplified version of the main model, focusing on class-level predictions with a custom severity-aware loss function. It consists of:

- **Feature Extractor**:
  - Same as the main model.
- **Classifier**:
  - Two fully connected layers with dropout (0.5) and a final classification layer (100 classes).

Parameter Count:

- **Total Parameters**: 5,918,628
- **Trainable Parameters**: 5,918,628

---

2. Training Dynamics

2.1 Main Assignment

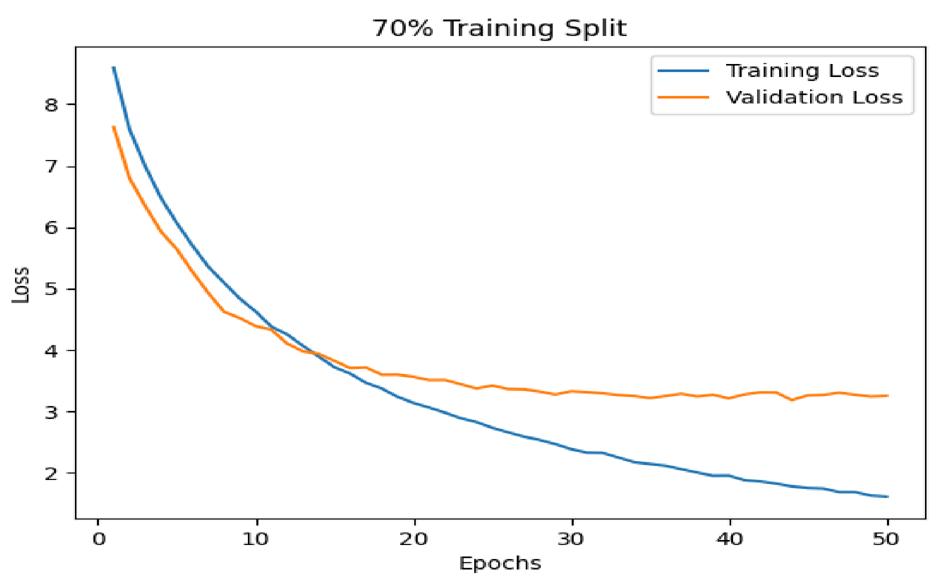
- **Training Splits**: 70%, 80%, 90%
- **Epochs**: 50
- **Batch Size**: 128
- **Optimizer**: Adam (lr=0.001)
- **LR Scheduling**: ReduceLROnPlateau (patience=5)
- **Loss Function**: Sum of cross-entropy losses for class, superclass, and group predictions.

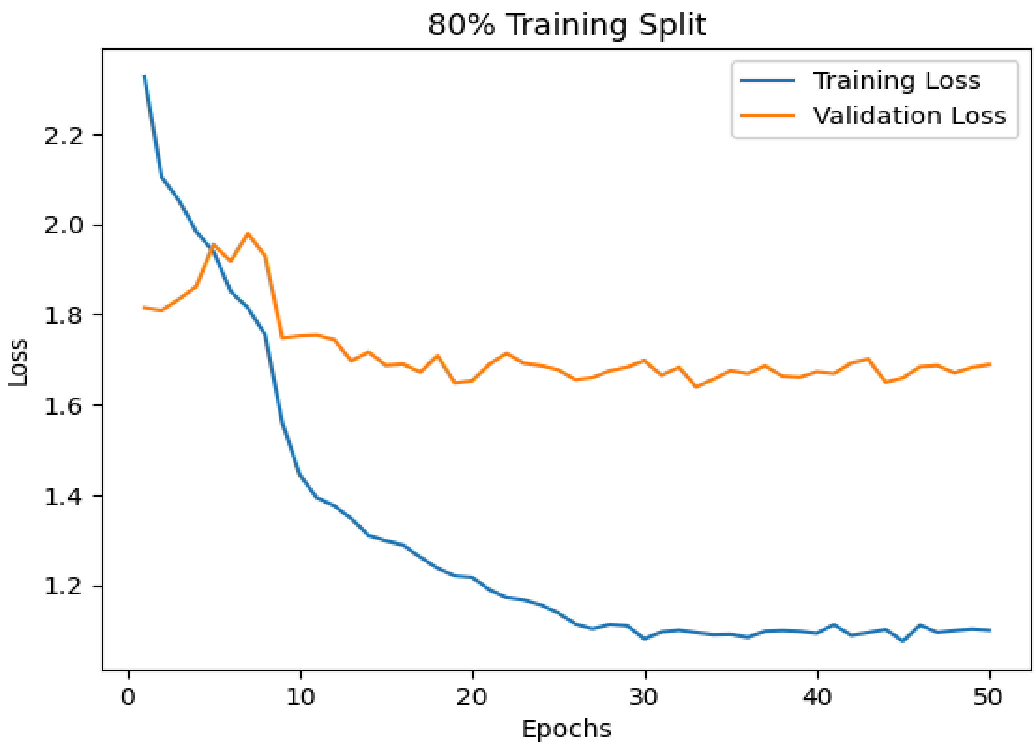
Training Loss Trends:

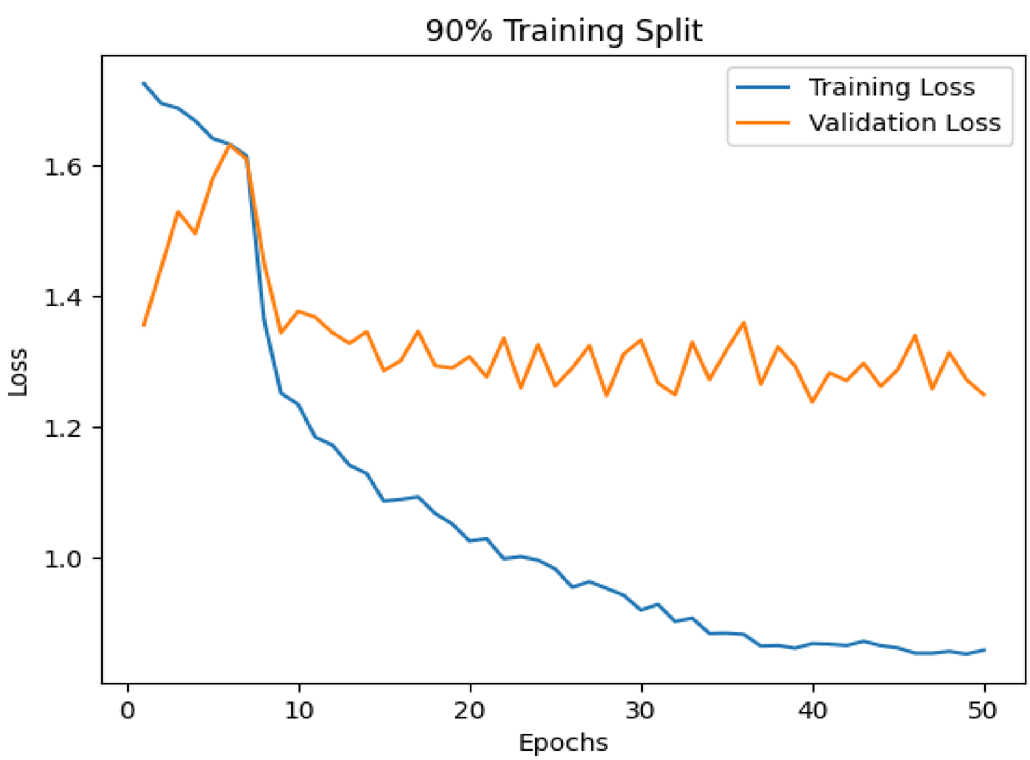
- **70% Split**: Loss decreases from 8.59 to 1.61.
- **80% Split**: Loss decreases from 2.33 to 1.10.
- **90% Split**: Loss decreases from 1.73 to 0.86.

Validation Loss Trends:

- **70% Split**: Loss decreases from 7.63 to 3.25.
- **80% Split**: Loss decreases from 1.81 to 1.69.
- **90% Split**: Loss decreases from 1.67 to 1.25.




80% Training Split
Training Loss
Validation Loss
Loss
Epochs
2.2
2.0
1.8
1.6
1.4
1.2
0
10
20
30
40
50


90% Training Split
Training Loss
Validation Loss
Loss
Epochs
1.6
1.4
1.2
1.0
0
10
20
30
40
50

2.2 Severity-Aware Implementation

- **Training Splits**: 70%, 80%, 90%
- **Epochs**: 50
- **Batch Size**: 128
- **Optimizer**: Adam (lr=0.001)
- **LR Scheduling**: ReduceLROnPlateau (patience=5)
- **Loss Function**: Custom severity-aware loss, weighting cross-entropy by error severity.

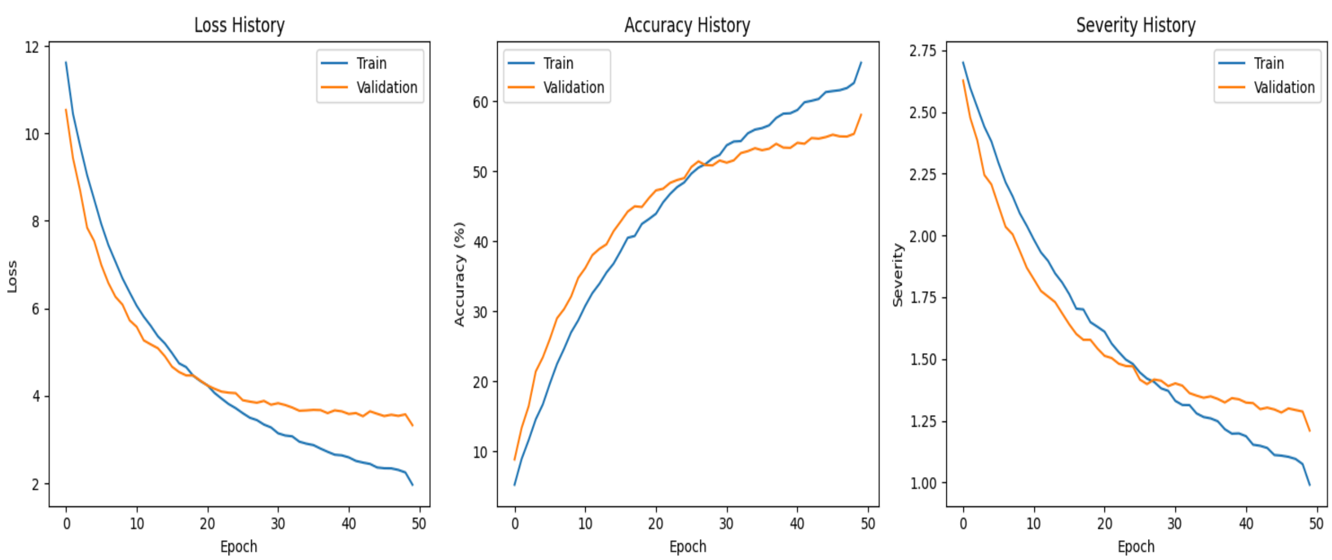
Training Loss Trends:

- **70% Split**: Loss decreases from 11.65 to 2.25.
- **80% Split**: Loss decreases from 2.96 to 1.51.
- **90% Split**: Loss decreases from 2.31 to 1.25.

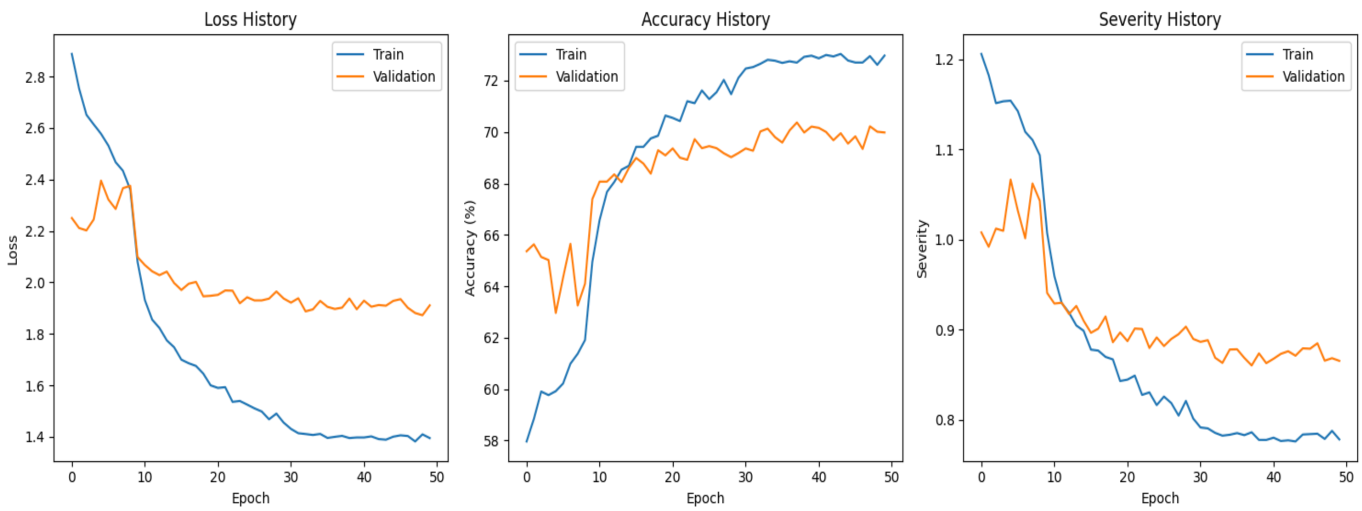
Validation Loss Trends:

- **70% Split**: Loss decreases from 10.37 to 3.57.
- **80% Split**: Loss decreases from 1.67 to 1.86.
- **90% Split**: Loss decreases from 1.67 to 1.43.

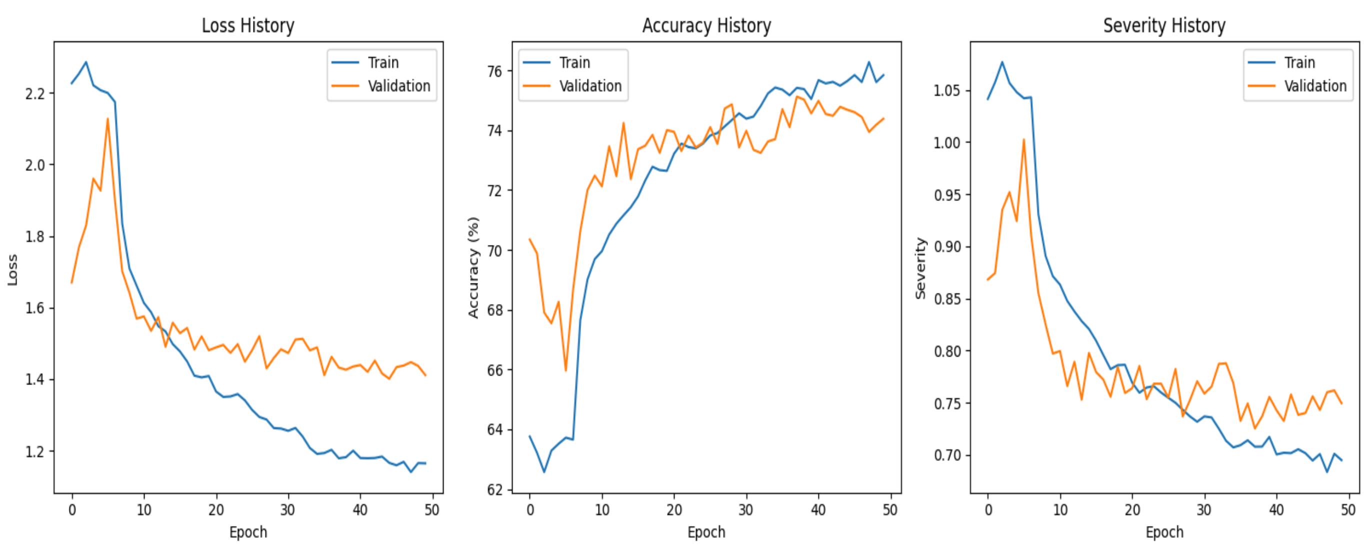
70% Split



80% Split



90% Split



3. Performance Metrics

3.1 Main Assignment

Training Set Results:

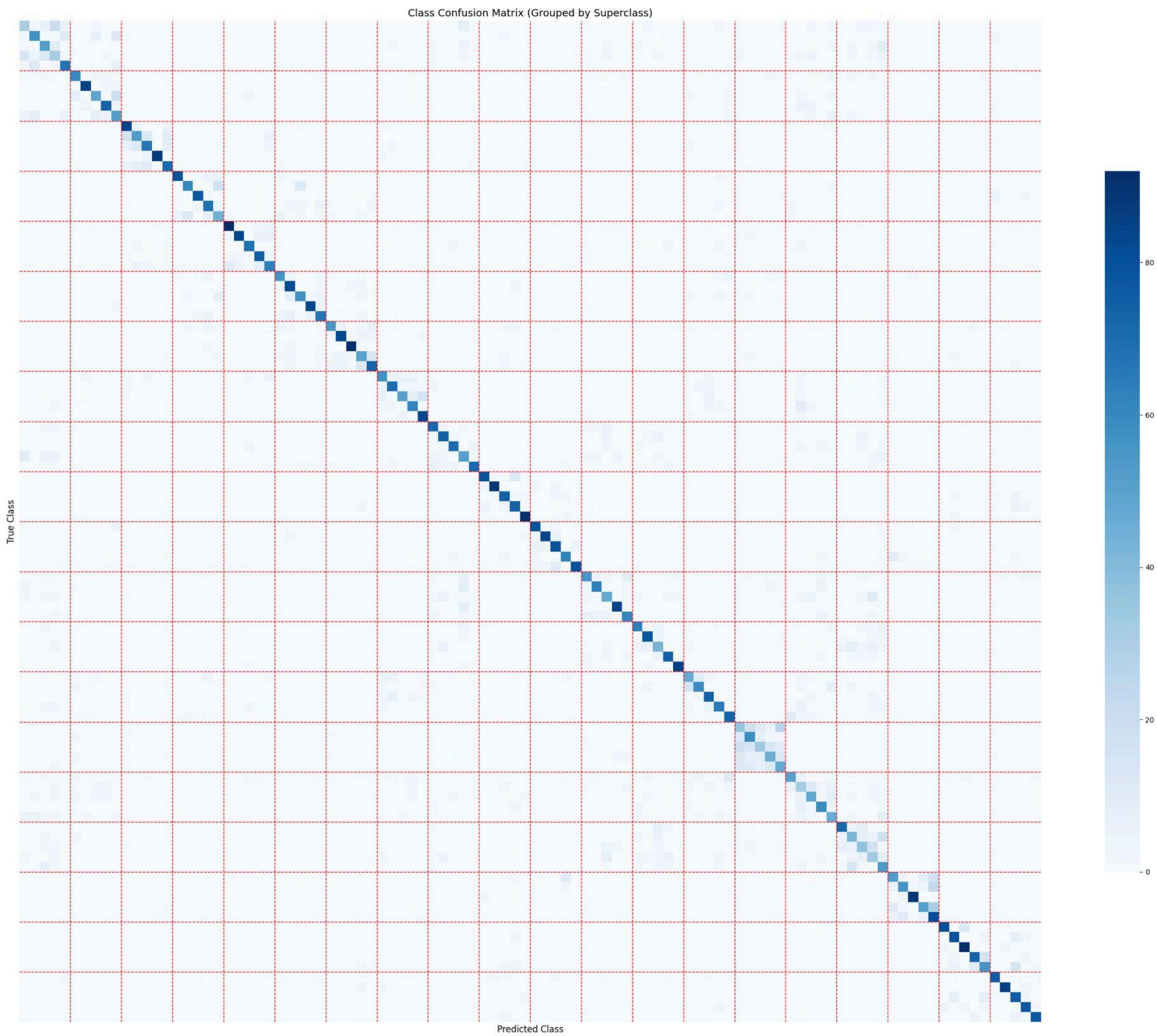
- **70% Split**:
  - Class Accuracy: 77.65%
  - Superclass Accuracy: 92.51%
  - Group Accuracy: 95.98%
- **80% Split**:
  - Class Accuracy: 83.63%
  - Superclass Accuracy: 95.59%
  - Group Accuracy: 98.06%
- **90% Split**:
  - Class Accuracy: 86.94%
  - Superclass Accuracy: 97.50%
  - Group Accuracy: 99.03%

Test Set Results:

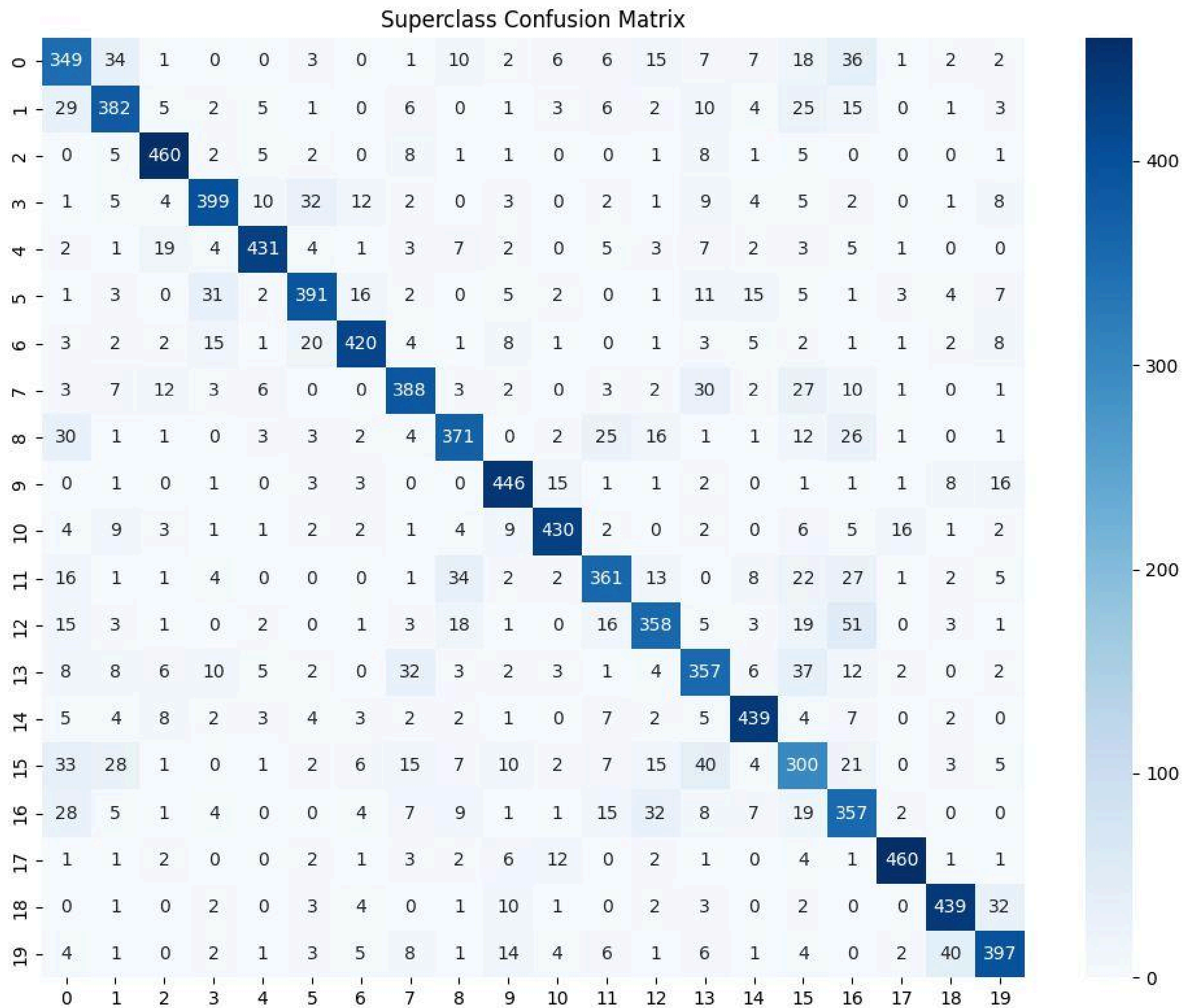
- **70% Split**:
  - Class Accuracy: 59.68%
  - Superclass Accuracy: 74.72%
  - Group Accuracy: 81.40%
- **80% Split**:
  - Class Accuracy: 64.61%
  - Superclass Accuracy: 78.68%
  - Group Accuracy: 84.49%
- **90% Split**:
  - Class Accuracy: 66.14%
  - Superclass Accuracy: 79.65%
  - Group Accuracy: 84.86%

Confusion Matrix

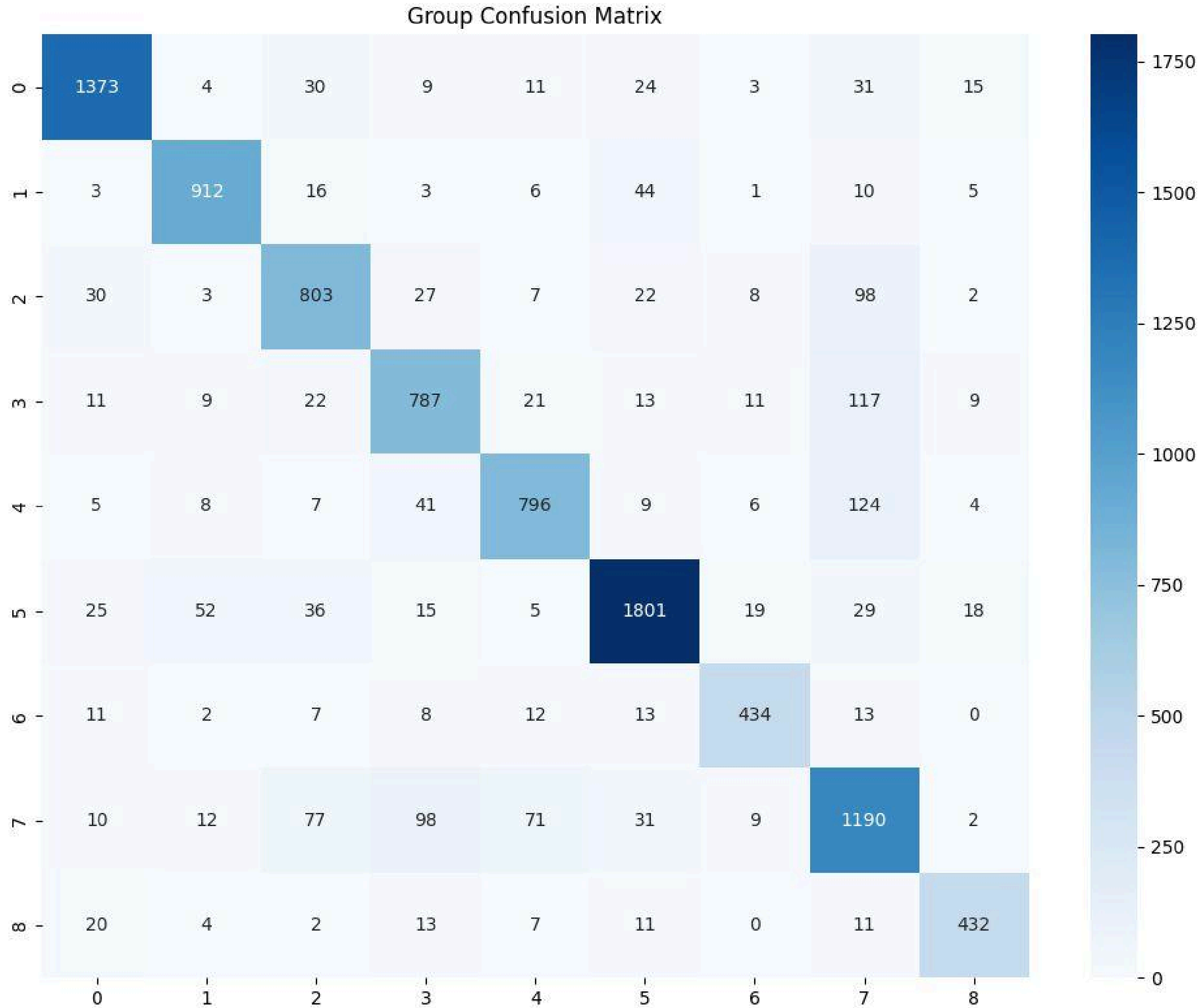
70% Split - Class Wise



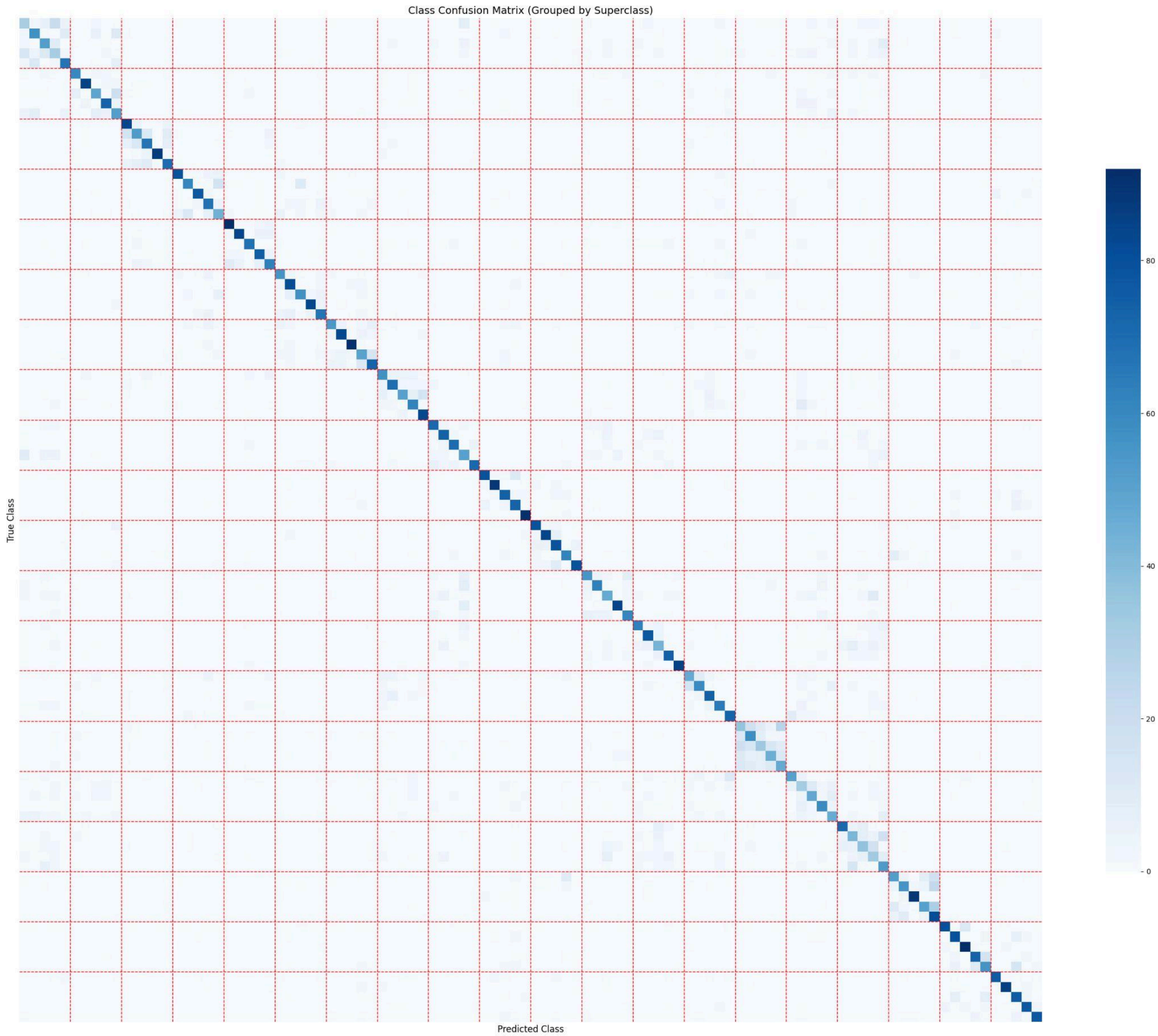
70% Split - Superclass Wise



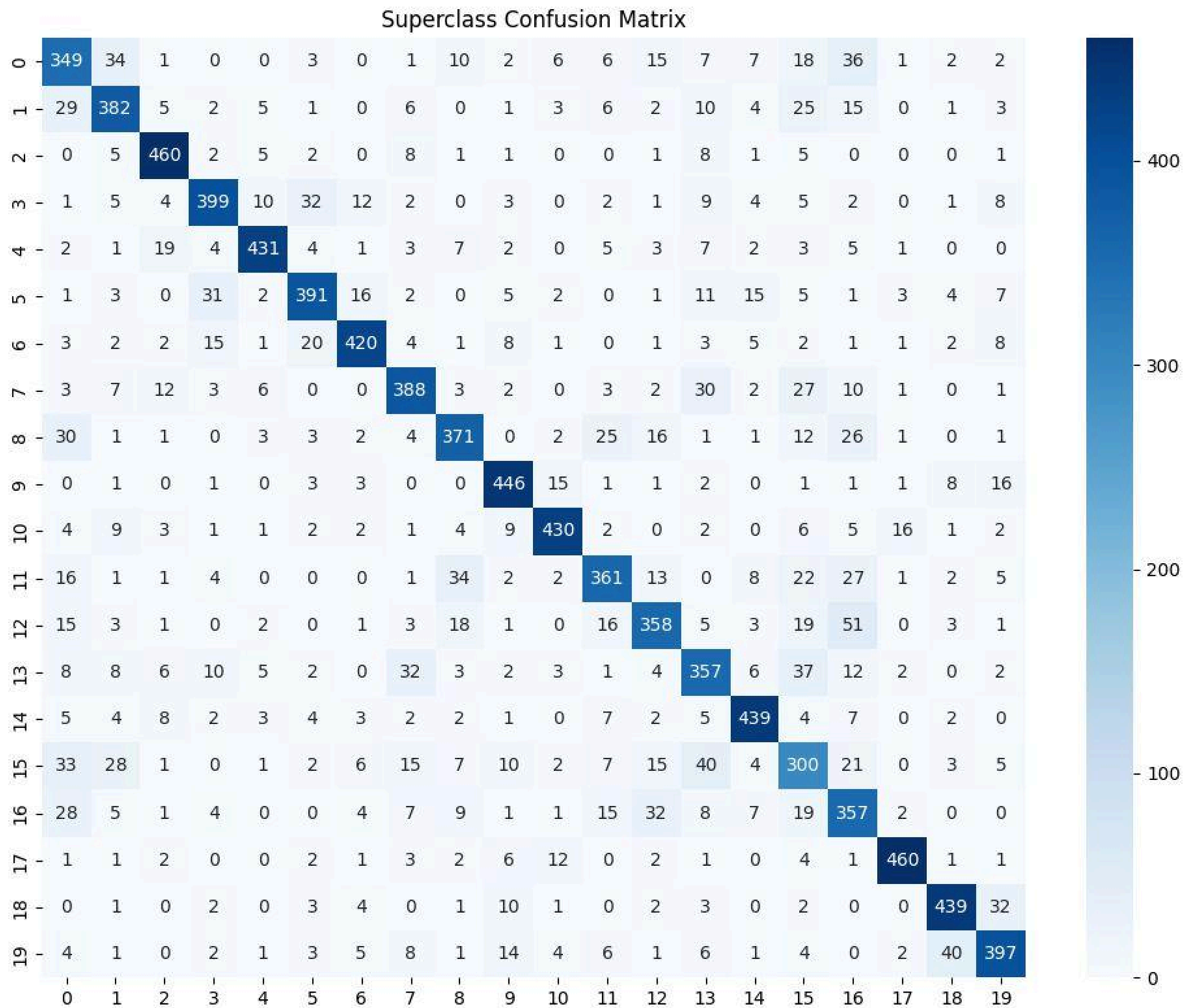
70% Split - Group Wise



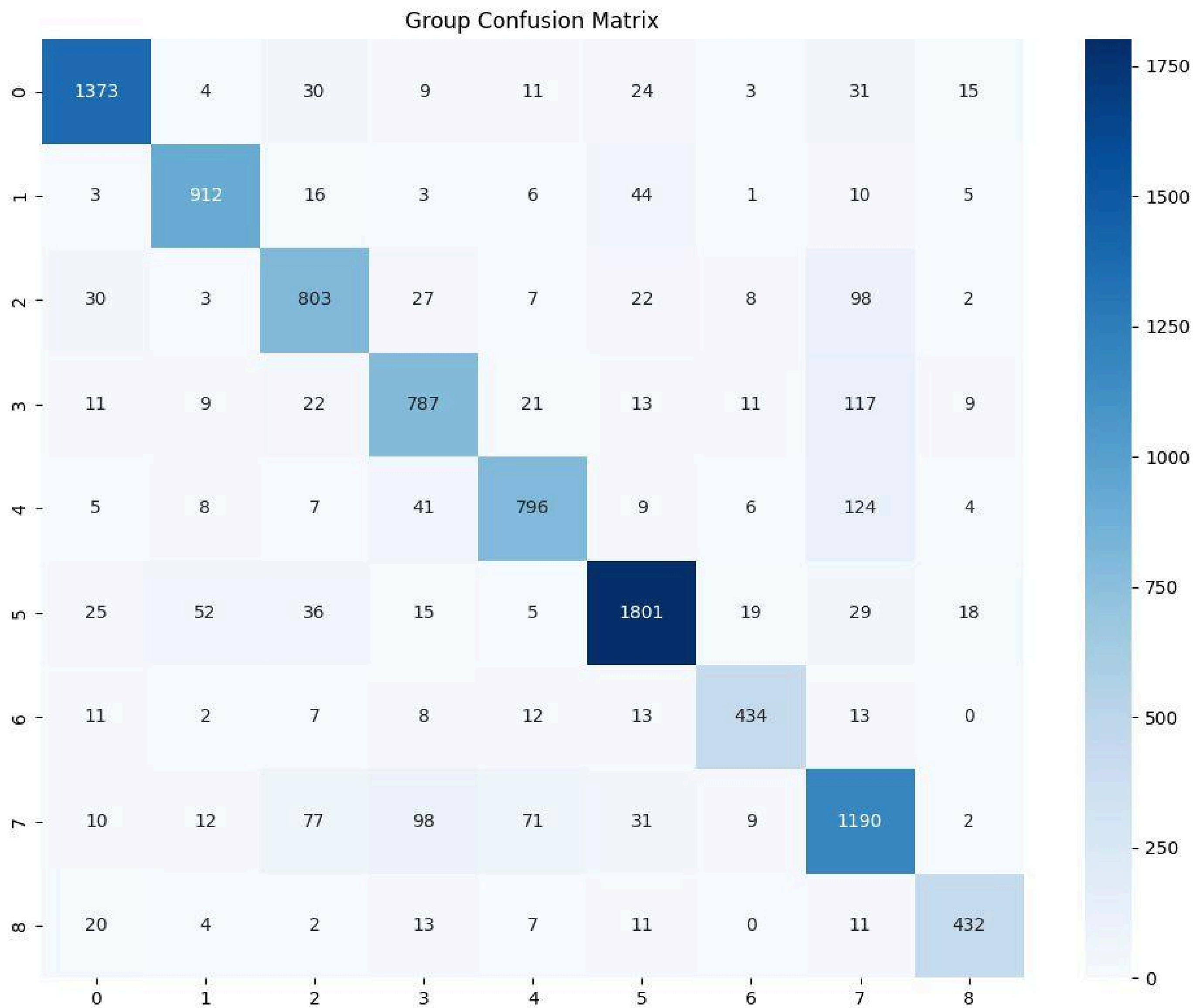
80% Split - Class Wise



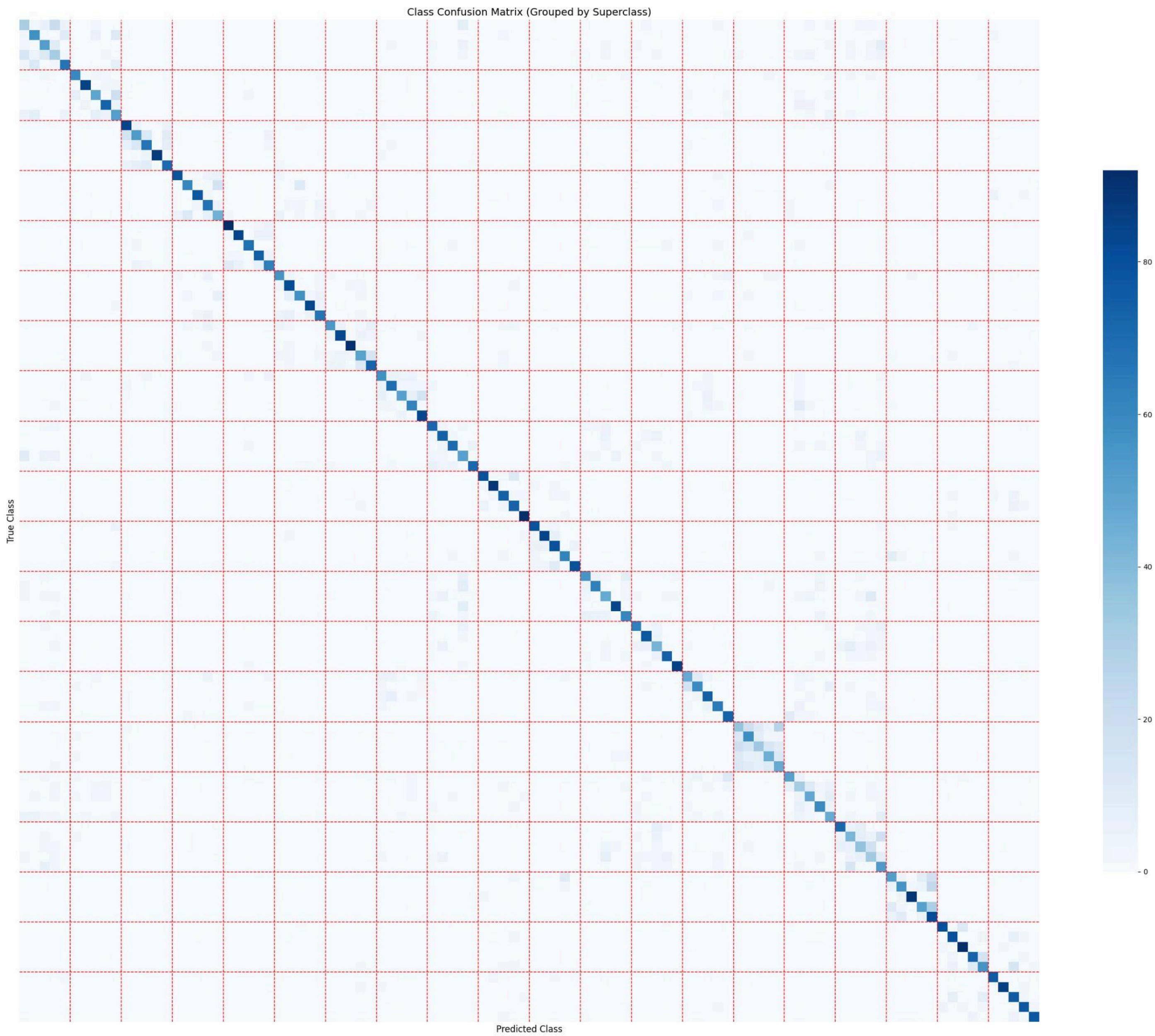
80% Split - Superclass Wise



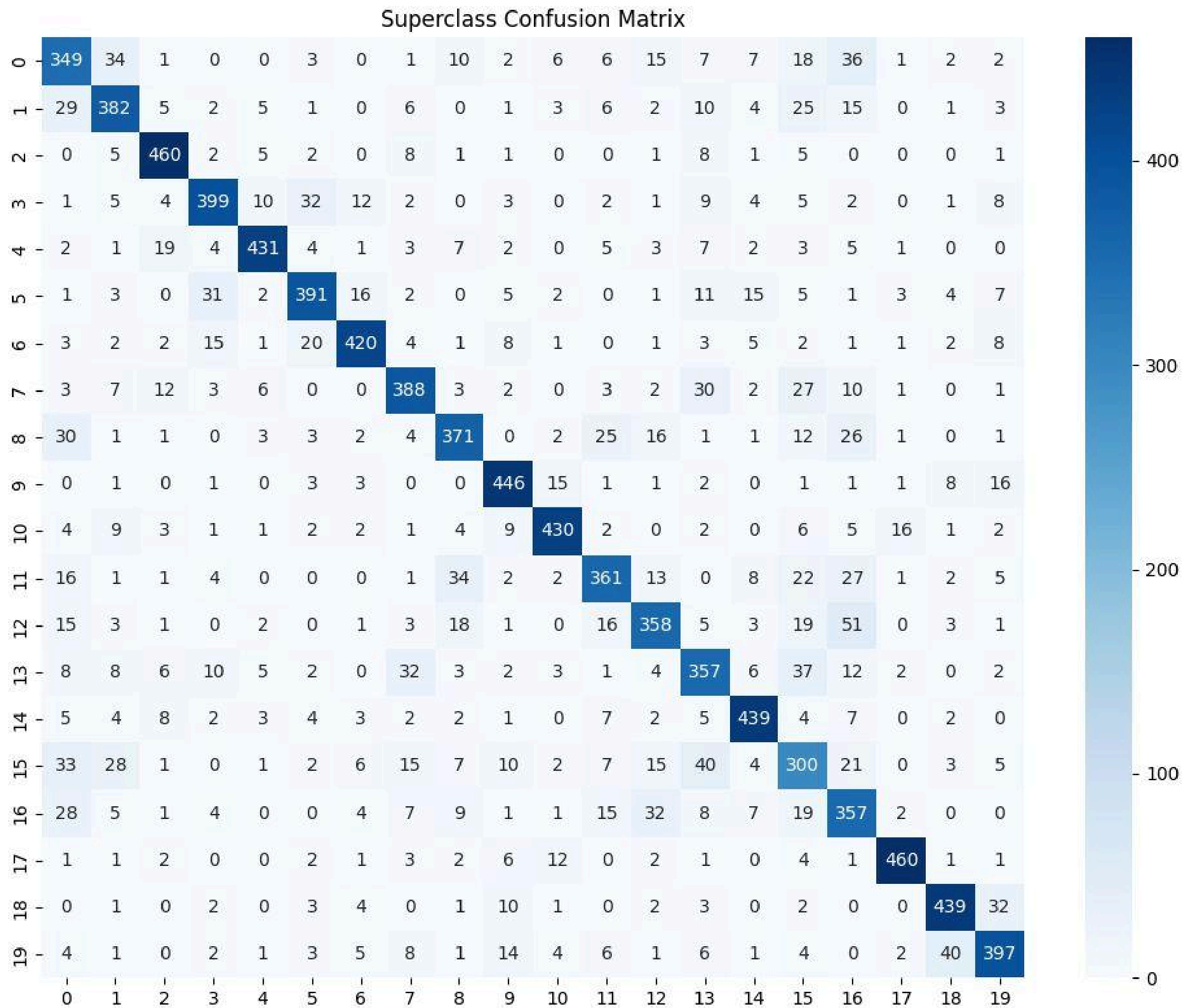
80% Split - Group Wise



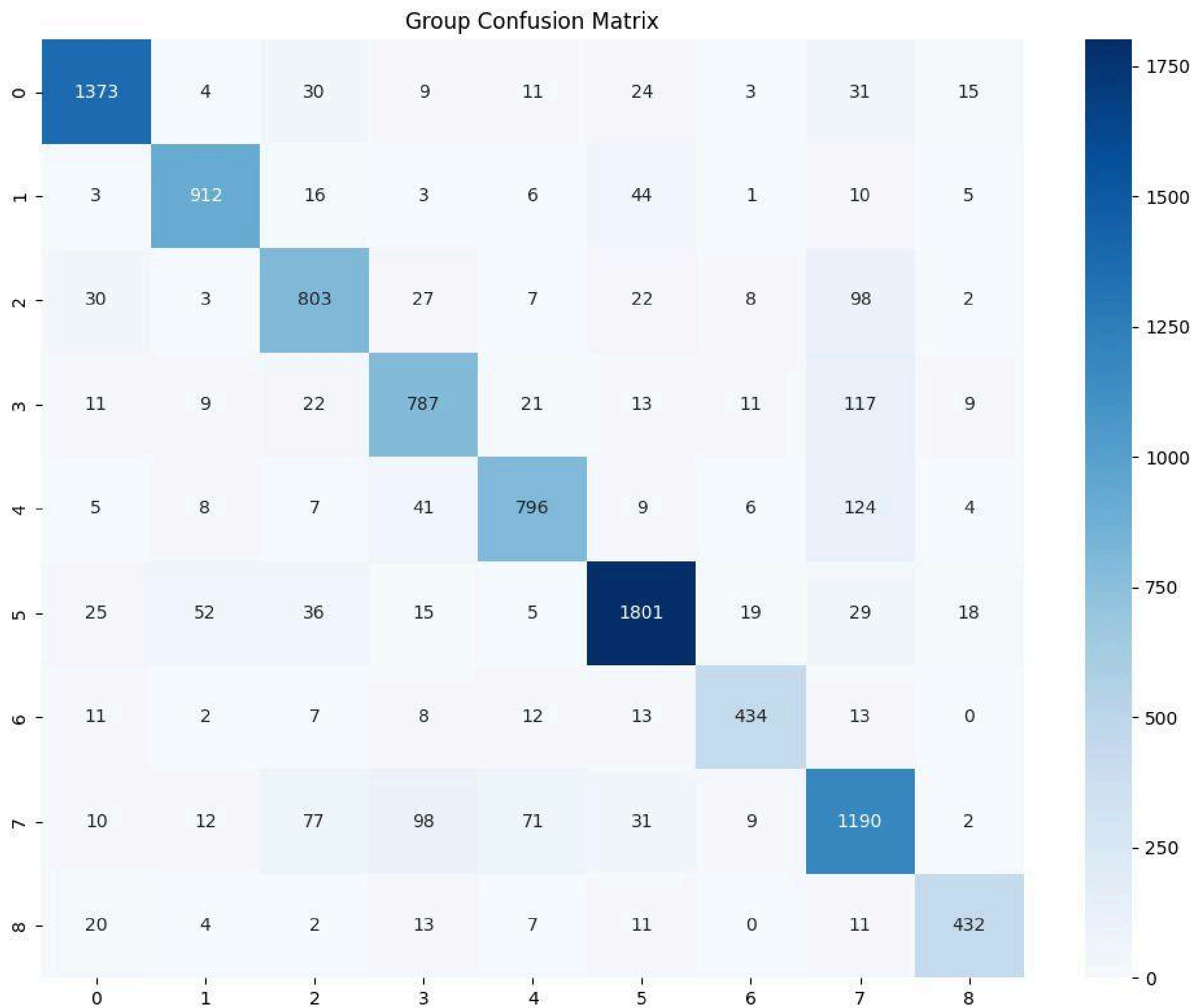
90% Split - Class Wise



90% Split - Superclass Wise



90% Split - Group Wise



3.2 Severity-Aware Implementation

Training Set Results:

- **70% Split**:
  - Accuracy: 62.41%
  - Average Severity: 1.0783
- **80% Split**:
  - Accuracy: 71.56%
  - Average Severity: 0.8173
- **90% Split**:
  - Accuracy: 74.76%
  - Average Severity: 0.7256

Test Set Results:

- **70% Split**:
  - Accuracy: 56.71%
  - Average Severity: 1.2389
- **80% Split**:
  - Accuracy: 62.34%
  - Average Severity: 1.0757
- **90% Split**:
  - Accuracy: 63.36%
  - Average Severity: 1.0499

---

4. Severity-Aware Implementation

4.1 Severity Matrix Design

The severity matrix assigns penalties based on the hierarchical relationships between classes:

- **Same Class**: Severity = 0
- **Same Superclass**: Severity = 1
- **Same Group**: Severity = 2
- **Different Group**: Severity = 3

4.2 Custom Loss Function

The **SeverityAwareLoss** function:

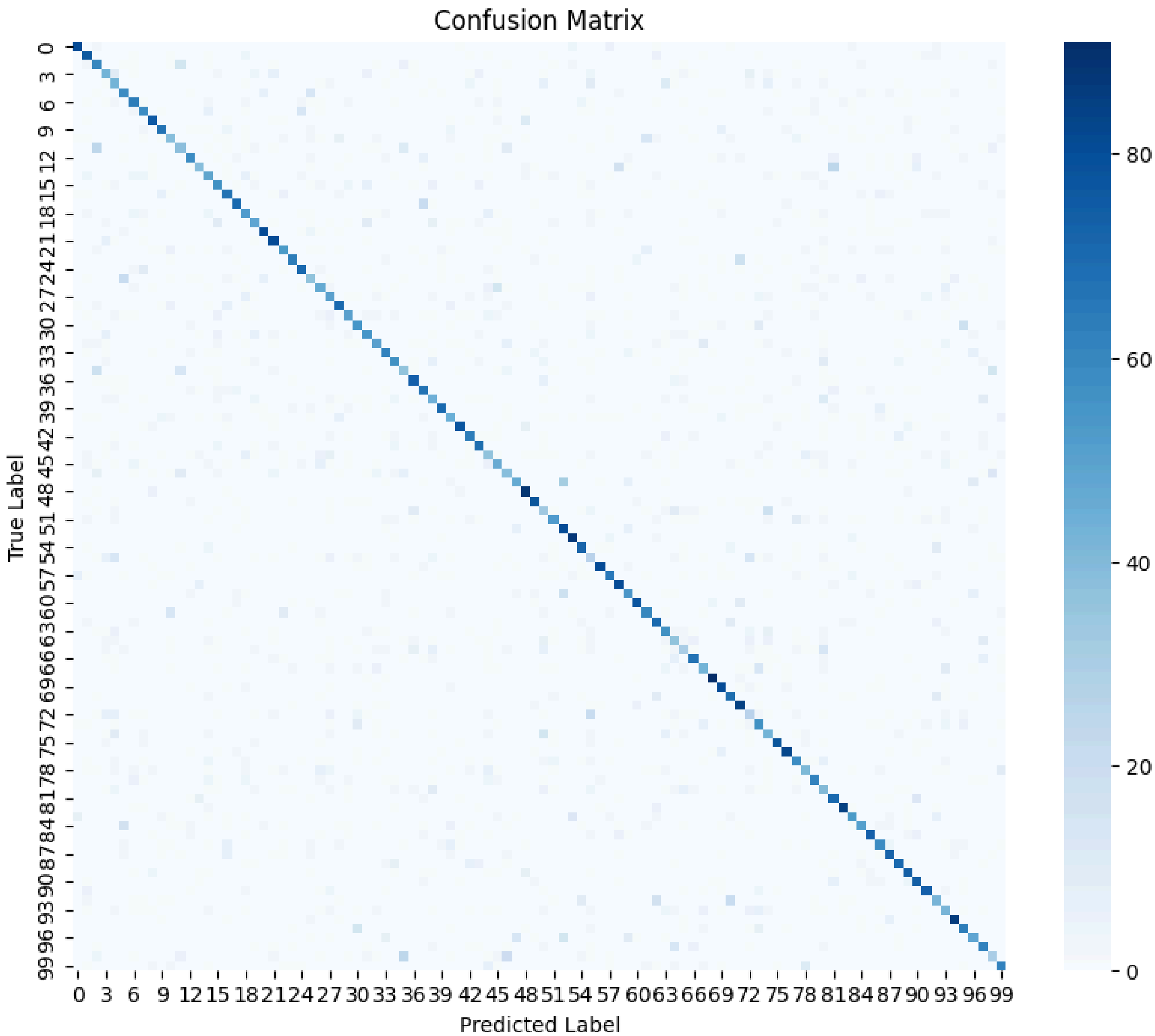
- Computes standard cross-entropy loss.
- Multiplies the loss by the severity weight for each misclassification.
- Encourages the model to make "less severe" errors.

4.3 Training Metrics

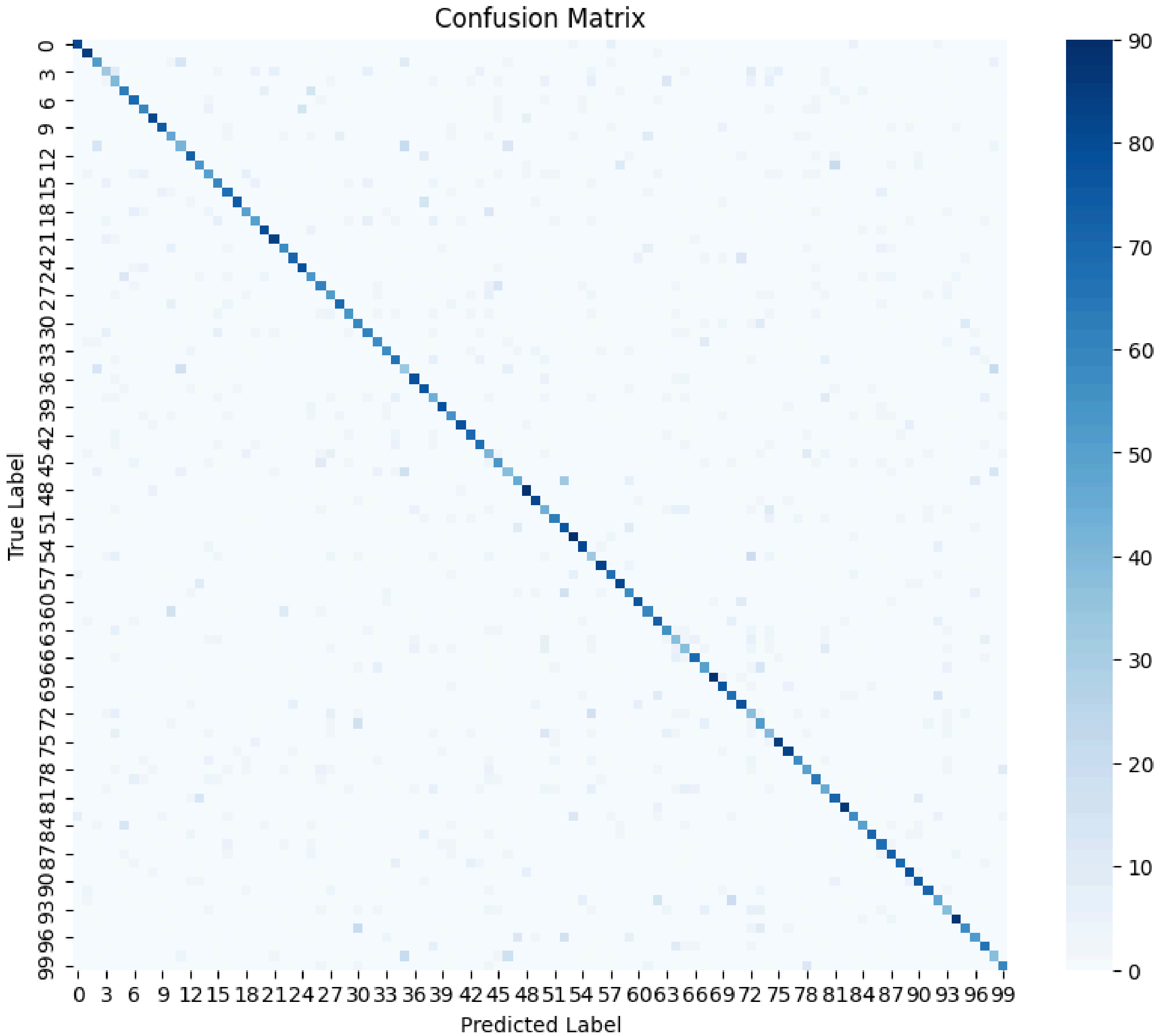
- **Severity Tracking**: Average severity per batch is tracked during training and validation.
- **Accuracy vs. Severity Trade-off**: The model balances accuracy and error severity, prioritizing less severe misclassifications.

Confusion Matrix for Bonus

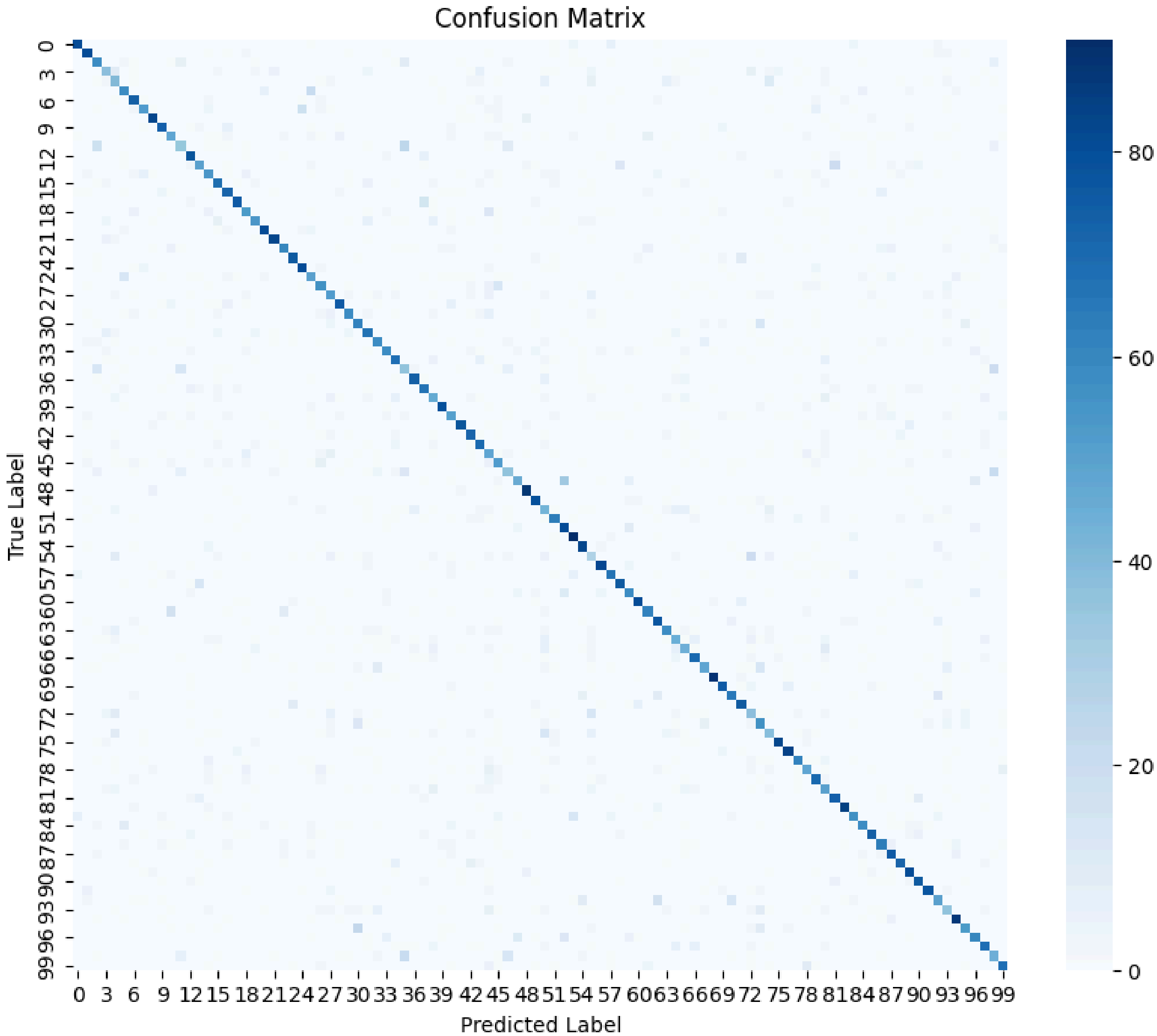
70% Split



80% Split



90% Split


Confusion Matrix
True Label
Predicted Label
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
0
20
40
60
80

5. Comparative Analysis

5.1 Accuracy

- **Main Model**: Higher accuracy across all splits (59.68%–66.14%).
- **Severity-Aware Model**: Lower accuracy (56.71%–63.36%) but with controlled error severity.

5.2 Error Severity

- **Main Model**: No control over error severity.
- **Severity-Aware Model**: Explicitly minimizes severe errors, achieving lower average severity (1.2389–1.0499).

5.3 Training Dynamics

- **Main Model**: Faster convergence due to multi-task learning.
- **Severity-Aware Model**: Slower convergence due to the complexity of the severity-aware loss.

---

6. Critical Observations

1. **Overfitting**:
   - The main model shows significant overfitting, especially with larger training splits (e.g., 90% split).
   - The severity-aware model exhibits less overfitting, likely due to the regularization effect of the severity penalty.
2. **Severity Matrix Limitations**:
   - Fixed severity weights may not capture the true semantic relationships between classes.
   - The matrix assumes a strict hierarchy, which may not always hold.
3. **Confusion Matrix Visualization**:
   - The main model's confusion matrices are sorted by superclass, but the severity-aware model lacks this feature.
4. **Training Time**:

- The severity-aware model requires more epochs to converge due to the additional complexity of the loss function.

---

7. Recommendations for Improvement

1. **Dynamic Severity Weights**:
   - Make the severity matrix learnable to adapt to the data.
2. **Advanced Architectures**:
   - Use pre-trained models (e.g., ResNet) as feature extractors.
3. **Enhanced Regularization**:
   - Add weight decay or dropout to the severity-aware model to reduce overfitting.
4. **Hierarchical Consistency Loss**:
   - Introduce a loss term to enforce consistency between class, superclass, and group predictions.
5. **Interactive Visualization**:
   - Develop interactive confusion matrices with tooltips for better error analysis.
6. **Hyperparameter Tuning**:
   - Perform a grid search for optimal learning rates, batch sizes, and severity weights.

Conclusion

The **Main Assignment** and **Severity-Aware Bonus Implementation** demonstrate two distinct approaches to hierarchical classification on CIFAR-100. The main model achieves higher accuracy but lacks control over error severity, while the severity-aware model trades some accuracy for more semantically meaningful errors. Future work should focus on improving the severity-aware loss and leveraging advanced architectures for better performance.