INFO6147
Deep Learning with PyTorch

Project 2 – Traffic Sign Classification

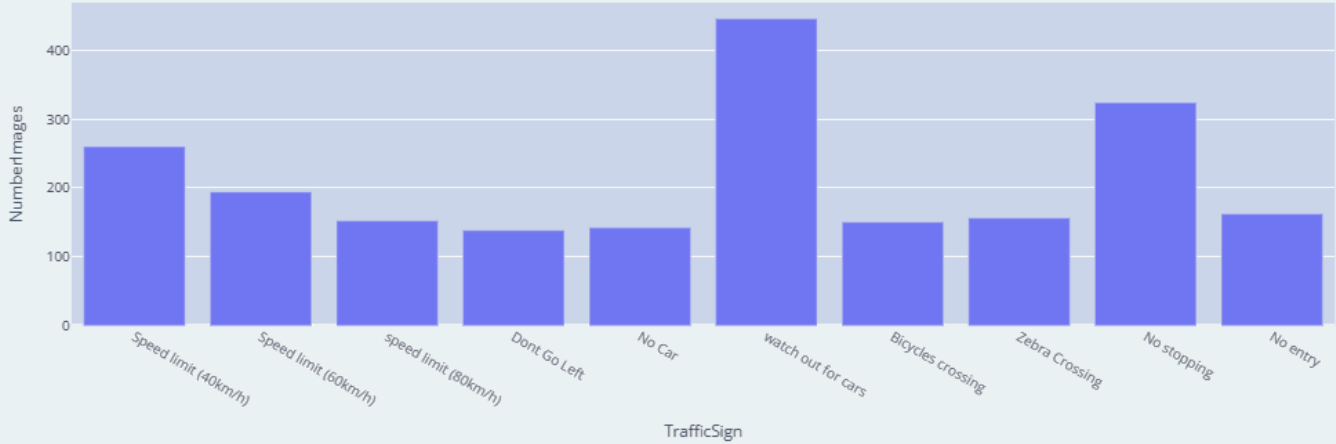
Student: Thai Kien Nguyen
Professor: Mohammed Yousefhussien

Content

1. Dataset Selection
2. Data Preprocessing
3. Model Selection and Training
4. Model Evaluation
5. Hyperparameter Tuning
6. Final results
7. Conclusion

1. Data Selection

- Traffic Sign Dataset: https://www.kaggle.com/datasets/ahemateja19bec1025/traffic-sign-dataset-classification/data
- Contains 58 classes, with 4170 training images and 1994 test images (sizes range from 80-300px).
- Includes traffic signs from China and Vietnam.
- Only 16 classes have more than 100 images.
- Selected the 10 classes with the highest number of images, ranging from 138 to 446 images per class.



2. Data Preprocessing

- Perform data transform steps:
  - Random brightness (20%) and contrast (20%) with ColorJitter
  - Random Rotate (15 degrees)
  - Random Resize (80-100% original size) and Crop (100x100)
  - Normalizing (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225] like ImageNet stats)
- Implemented a PyTorch DataSet to load images and apply an oversampling technique for the training images.
- Split the dataset into training (3568 images), validation (892 images), and test sets (542 images).
- Initially, tried various parameters and architectures, including a function to ignore or enhance dark images before loading the dataset (normalized images appeared too dark). However, after extensive optimization, these were disabled due to achieving good results without them.
- Convert DataSet to DataLoader

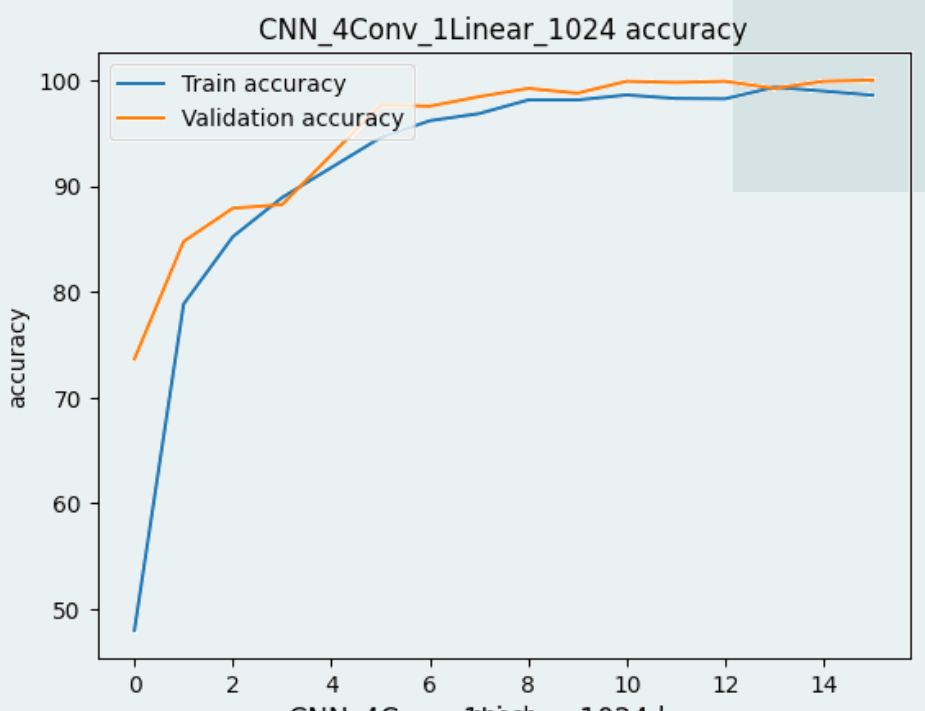
3. Model Selection and Training

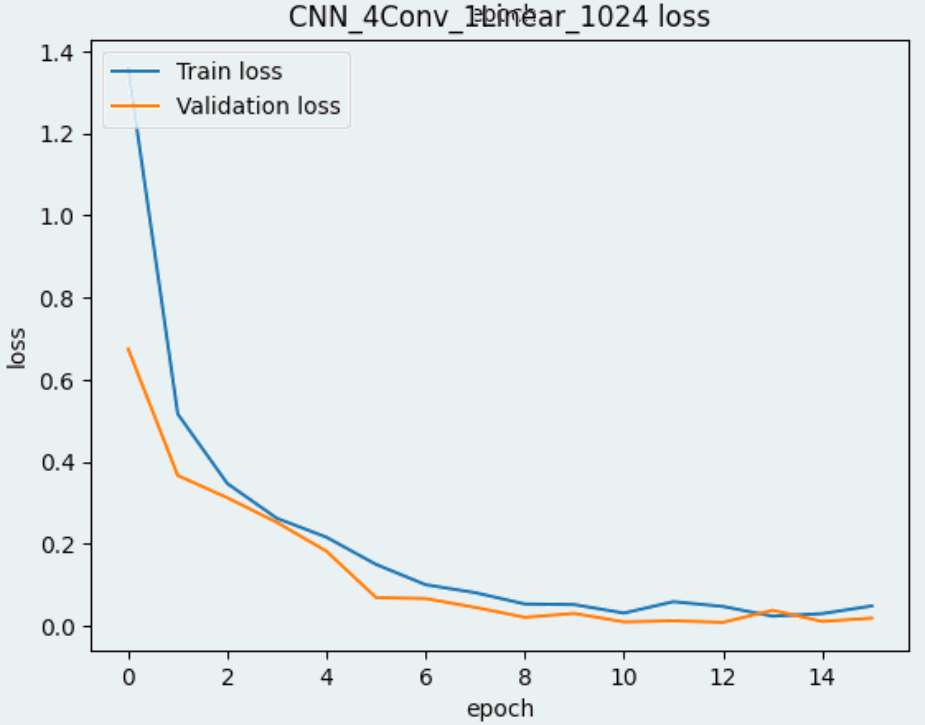
- CNN Model: (partly tuned)
  - Input size: 3x100x100
  - 4 Conv2d Blocks
    - Conv2D
    - BatchNorm2d (Tuning phase)
    - ReLu
    - MaxPool2D
    - Dropout
  - Flatten
  - Linear Layer with 1024 units
  - Output Layer with 10 classes

```
CNN_Model                                [32, 10]                  --
├─ModuleList: 1-1                        --                        --
│    └─Sequential: 2-1                   [32, 32, 50, 50]          --
│    │    └─Conv2d: 3-1                  [32, 32, 100, 100]        896
│    │    └─ReLU: 3-2                    [32, 32, 100, 100]        --
│    │    └─MaxPool2d: 3-3               [32, 32, 50, 50]          --
│    │    └─Dropout: 3-4                 [32, 32, 50, 50]          --
│    └─Sequential: 2-2                   [32, 64, 25, 25]          --
│    │    └─Conv2d: 3-5                  [32, 64, 50, 50]          18,496
│    │    └─ReLU: 3-6                    [32, 64, 50, 50]          --
│    │    └─MaxPool2d: 3-7               [32, 64, 25, 25]          --
│    │    └─Dropout: 3-8                 [32, 64, 25, 25]          --
│    └─Sequential: 2-3                   [32, 128, 12, 12]         --
│    │    └─Conv2d: 3-9                  [32, 128, 25, 25]         73,856
│    │    └─ReLU: 3-10                   [32, 128, 25, 25]         --
│    │    └─MaxPool2d: 3-11              [32, 128, 12, 12]         --
│    │    └─Dropout: 3-12                [32, 128, 12, 12]         --
│    └─Sequential: 2-4                   [32, 256, 6, 6]           --
│    │    └─Conv2d: 3-13                 [32, 256, 12, 12]         295,168
│    │    └─ReLU: 3-14                   [32, 256, 12, 12]         --
│    │    └─MaxPool2d: 3-15              [32, 256, 6, 6]           --
│    │    └─Dropout: 3-16                [32, 256, 6, 6]           --
├─Sequential: 1-2                        [32, 10]                  --
│    └─Flatten: 2-5                      [32, 9216]                --
│    └─Linear: 2-6                       [32, 1024]                9,438,208
│    └─ReLU: 2-7                         [32, 1024]                --
│    └─Linear: 2-8                       [32, 10]                  10,250
==========================================================================
Total params: 9,836,874
Trainable params: 9,836,874
Non-trainable params: 0
Total mult-adds (Units.GIGABYTES): 4.91
```

3. Model Selection and Training

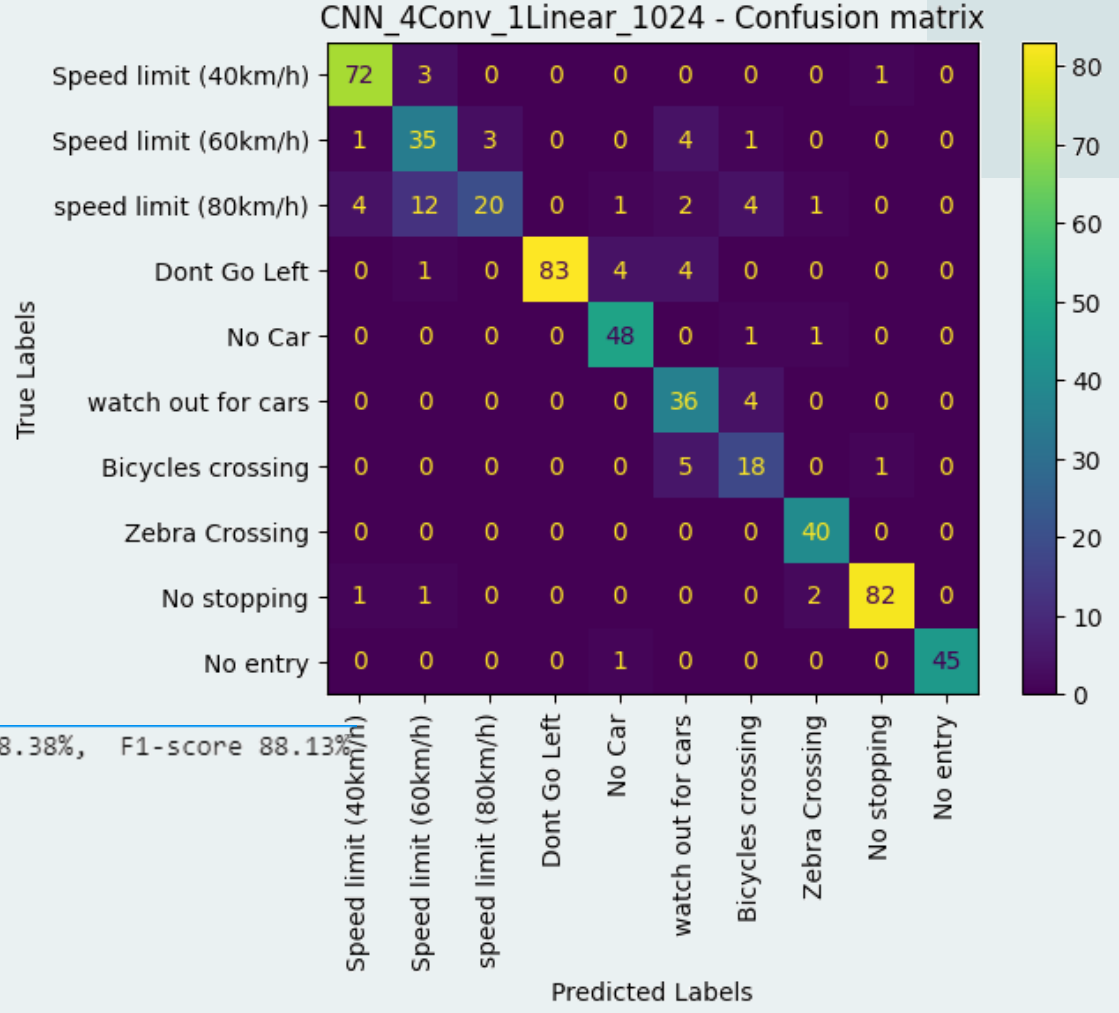
- Training process:
  - Train set + validation set
  - Epoch max: 20 (Early Stopping, restore best model at epoch 16)
  - Batch Size: 32
  - Learning rate: 0.001
  - Optimizer: Adam
  - Loss function: CrossEntropyLoss
  - Early Stopping: val_loss, patience=3
  - Environment: Local machine
  - Device: GPU
  - Training time: 113s





4. Model Evaluation

- Metrics:
  - Accuracy
  - Precision
  - Recall
  - F1-score
- Confusion Matrix



CNN_4Conv_1Linear_1024: Accuracy 88.38%, Precision 89.38%, Recall 88.38%, F1-score 88.13%

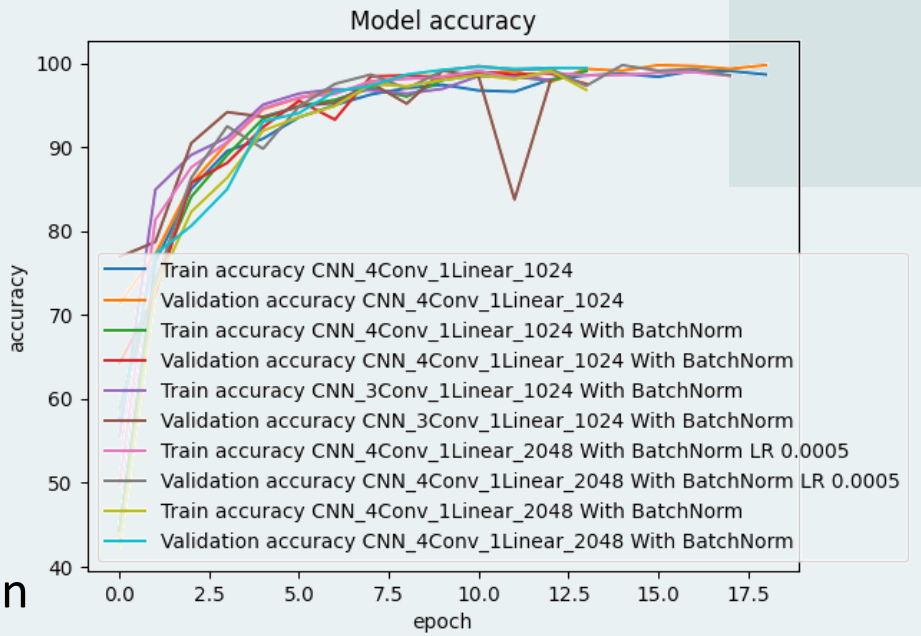
Detail report:

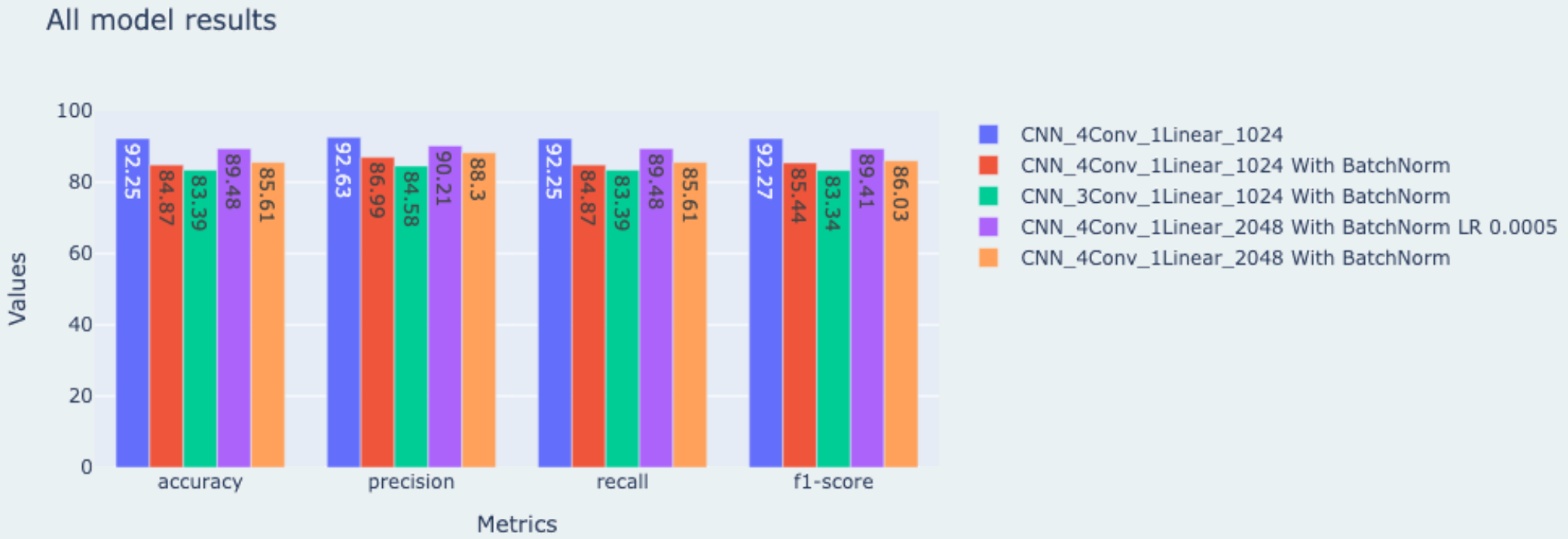
	precision	recall	f1-score	support
Speed limit (40km/h)	0.92	0.95	0.94	76
Speed limit (60km/h)	0.67	0.80	0.73	44
speed limit (80km/h)	0.87	0.45	0.60	44
Dont Go Left	1.00	0.90	0.95	92
No Car	0.89	0.96	0.92	50
watch out for cars	0.71	0.90	0.79	40
Bicycles crossing	0.64	0.75	0.69	24
Zebra Crossing	0.91	1.00	0.95	40
No stopping	0.98	0.95	0.96	86
No entry	1.00	0.98	0.99	46
accuracy			0.88	542
macro avg	0.86	0.86	0.85	542
weighted avg	0.89	0.88	0.88	542

4. Model Evaluation

5. Hyperparameter Tuning

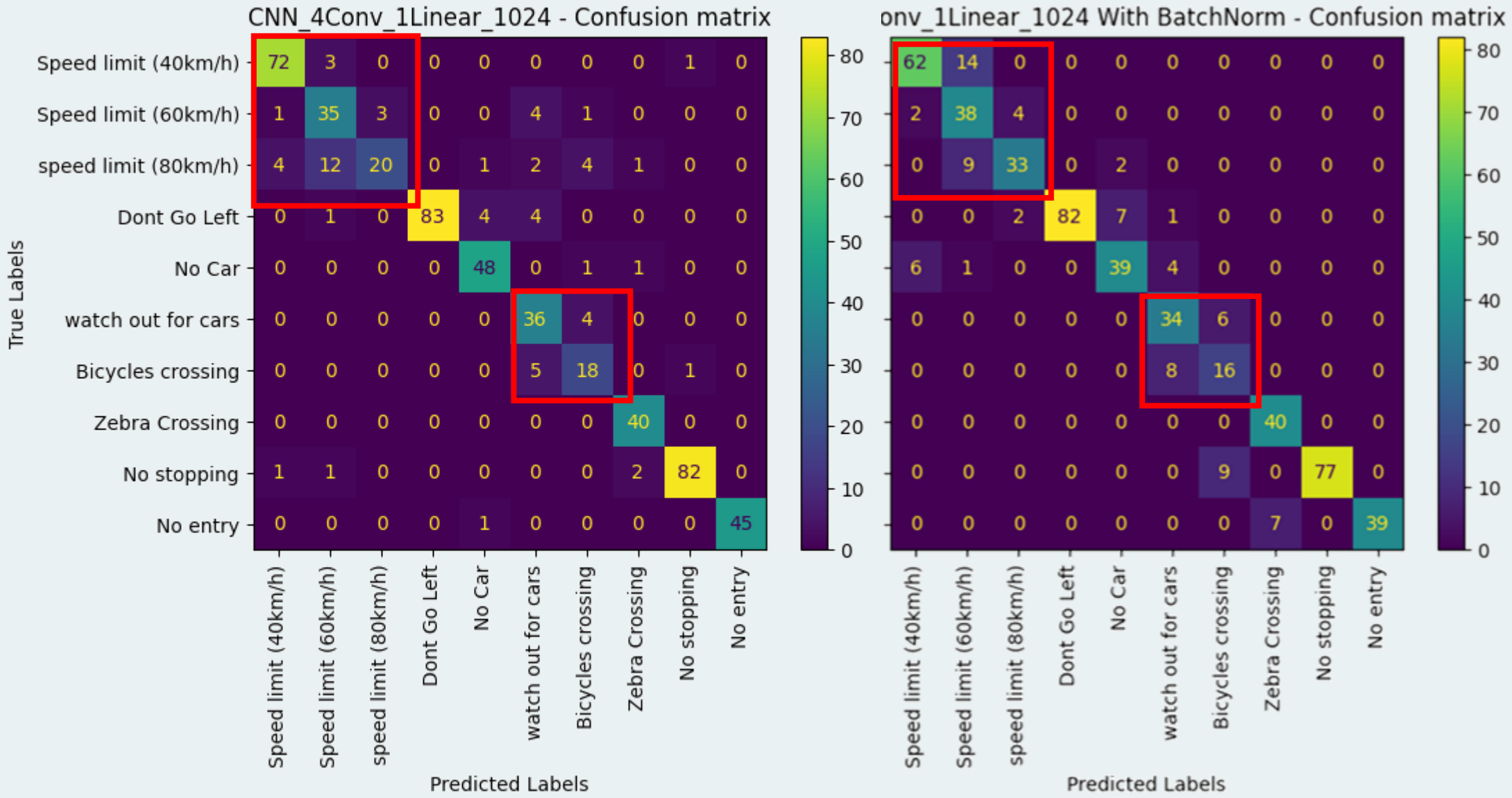
- Hyperparameter Tuning
  - Batch size, Learning Rate, Epoch (with EarlyStopping)
  - Weight Decay
- Tuning data augmentation config, preprocess images
- Try different model architectures or add Batch Normalization





6. Final results

- Confusion Matrix of all classes



6. Final results

- Similar traffic signs could result in misclassifying (same background color, border color, patterns or shape)

Group 1: Watch out for cars and Bicycles crossing
Group 2: Speed limit signs

6. Final results

- Model struggling to predict combined traffic signs

6. Final results

- All models performed well with high accuracy, precision, recall

7. Conclusion

- **Challenges and Experiences:**
  - Imbalanced datasets often require careful data preprocessing and cleaning.
  - High accuracy or other metrics do not necessarily imply a high classification rate for each class.
  - Similar class images need special treatment.
  - For image classification, object segmentation is crucial to accurately classify the main object.
  - Hyperparameter tuning consumes a significant amount of time.
  - There is a substantial gap between model development and real-world application, such as the need for advanced techniques to detect and crop valid traffic signs, classification of partially covered images or those in adverse weather conditions, and handling situations where the model is not confident in its predictions.

7. Conclusion

- **Future Improvements:**
  - Utilize the Canada Traffic Sign Dataset, possibly by manually collecting dashcam video data.
  - Expand the model to work with various traffic sign types.
  - Visualize a map highlighting the important regions in the image using Grad-Cam or other techniques to improve classification performance of similar traffic signs.
  - Implement real-time traffic sign classification using mobile or embedded system cameras.
  - Combine multiple models for object segmentation, anomaly detection (e.g., road obstacles), and integrate data from weather and light sensors to provide warnings or advice to drivers.
  - Enhance model robustness and security.

Thank you

References:

- https://www.learnpytorch.io/
- https://www.kaggle.com/code/sachinsarkar/traffic-sign-recognition-using-pytorch-and-cnn

Github: https://github.com/thkien1990/pytorchproject