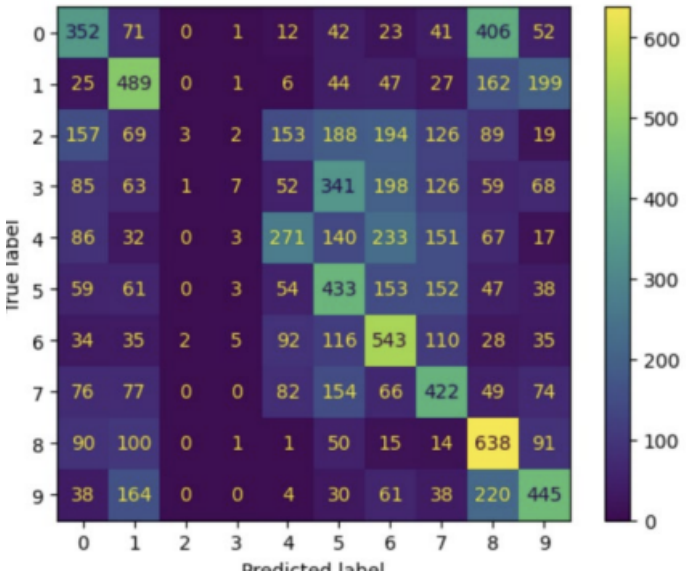
1. Image Classification with CNN (PyTorch Lightning)

Model Overview

- **Input**: A dataset consisting of images from **10 categories**—plane, car, bird, cat, deer, dog, frog, horse, ship, and truck.
- **Output**: The predicted class label for each image and its **confidence score** (probability).

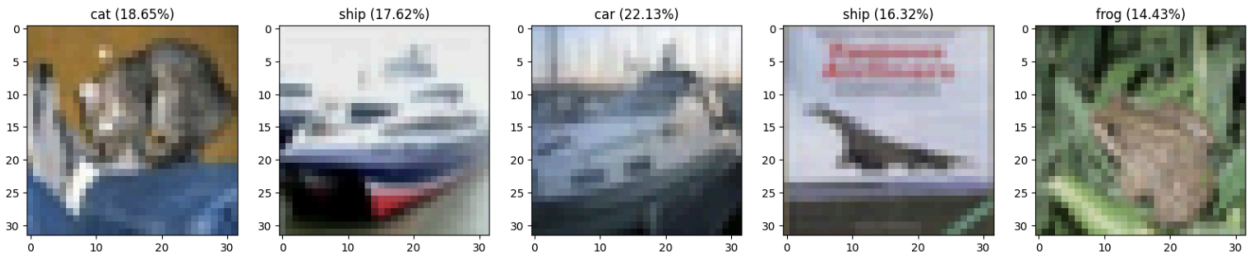
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN                                      [32, 10]                  --
├─Conv2d: 1-1                            [32, 6, 28, 28]           456
├─MaxPool2d: 1-2                         [32, 6, 14, 14]           --
├─Conv2d: 1-3                            [32, 16, 10, 10]          2,416
├─MaxPool2d: 1-4                         [32, 16, 5, 5]            --
├─Linear: 1-5                            [32, 120]                 48,120
├─Linear: 1-6                            [32, 84]                  10,164
├─Linear: 1-7                            [32, 10]                  850
├─Softmax: 1-8                           [32, 10]                  --
==========================================================================================
Total params: 62,006
Trainable params: 62,006
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 21.06
==========================================================================================
Input size (MB): 0.39
Forward/backward pass size (MB): 1.67
Params size (MB): 0.25
Estimated Total Size (MB): 2.31
==========================================================================================
```

- **Hardware Requirements**: The model primarily runs on **GPU (CUDA:0)** for faster computation. If a GPU is unavailable, the code automatically switches to the **CPU**.
- **Data Statistics**: The model evaluation includes **precision, recall, and F1-score** as key performance metrics.
- **Learning Curve**: The model tracks accuracy and loss throughout training and validation to visualize learning progression.
- **Metrics**: **Softmax activation** is applied in the final layer to compute class probabilities and determine the most likely label.
- **Demo Result**: Includes a statistical evaluation of the model's performance and sample predictions to illustrate classification accuracy.



Key Functions & Features (Cheat Sheet)

- **Dataset**:
  - The dataset consists of **60,000 images** divided into:
    - **40,000** for training
    - **10,000** for validation
    - **10,000** for testing
- **Training Process**:
  - Utilizes **PyTorch Lightning Trainer**
  - Runs for a **maximum of 1 epoch**
  - Records **training and validation loss & Accuracy**
- **Evaluation & Testing**:
  - The trained model is tested on the **test dataset**
  - Results are assessed using:
    - **Classification Precision**
    - **Confusion Matrix** (to visualize class-wise accuracy)
- **Final Output**:
  - The model's predictions are demonstrated using the **first 5 images** from the test set to visually represent its classification capabilities.



2. Image Classification with EfficientNetV2s (PyTorch Lightning)

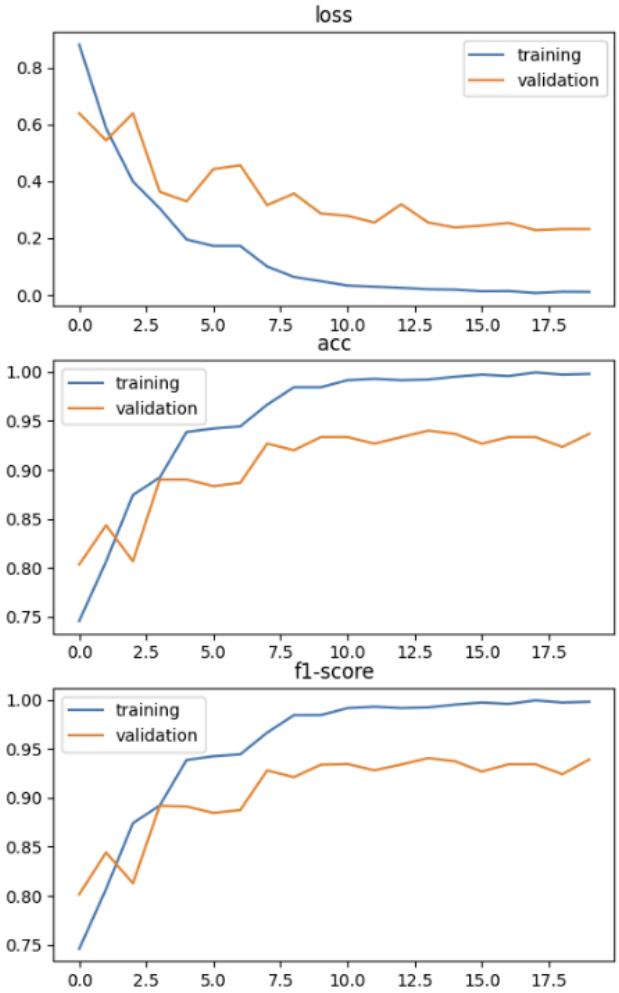
Model Overview

- **Input**: A dataset containing images of **10 animal classes**—butterfly, cat, chicken, cow, dog, elephant, horse, sheep, spider, and squirrel.
- **Output**: The predicted class for each image and a **confidence score** (probability).

```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [64, 24, 112, 112]             648
       BatchNorm2d-2        [64, 24, 112, 112]              48
              SiLU-3        [64, 24, 112, 112]               0
            Conv2d-4        [64, 24, 112, 112]           5,184
       BatchNorm2d-5        [64, 24, 112, 112]              48
              SiLU-6        [64, 24, 112, 112]               0
   StochasticDepth-7        [64, 24, 112, 112]               0
       FusedMBConv-8        [64, 24, 112, 112]               0
            Conv2d-9        [64, 24, 112, 112]           5,184
      BatchNorm2d-10        [64, 24, 112, 112]              48
             SiLU-11        [64, 24, 112, 112]               0
  StochasticDepth-12        [64, 24, 112, 112]               0
      FusedMBConv-13        [64, 24, 112, 112]               0
           Conv2d-14          [64, 96, 56, 56]          20,736
      BatchNorm2d-15          [64, 96, 56, 56]             192
             SiLU-16          [64, 96, 56, 56]               0
           Conv2d-17          [64, 48, 56, 56]           4,608
      BatchNorm2d-18          [64, 48, 56, 56]              96
      FusedMBConv-19          [64, 48, 56, 56]               0
           Conv2d-20         [64, 192, 56, 56]          82,944
      BatchNorm2d-21         [64, 192, 56, 56]             384
             SiLU-22         [64, 192, 56, 56]               0
...
Forward/backward pass size (MB): 20629.03
Params size (MB): 77.02
Estimated Total Size (MB): 20742.80
```

- **Hardware Requirements**: The model primarily runs on **CUDA:0 (GPU)** for efficient training. If a GPU is unavailable, it falls back to the **CPU**. The model used for classification is **LitEfficientNetV2**.
- **Data Statistics**:
    - Total **2000 images**, split into:
        - **1400** for training

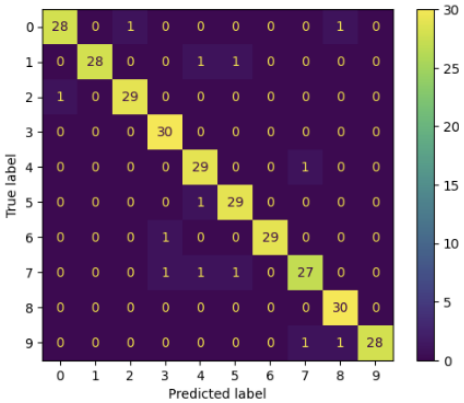
        - **300** for validation
        - **300** for testing
- **Learning Curve**: Tracks accuracy and loss progression across training and validation phases.

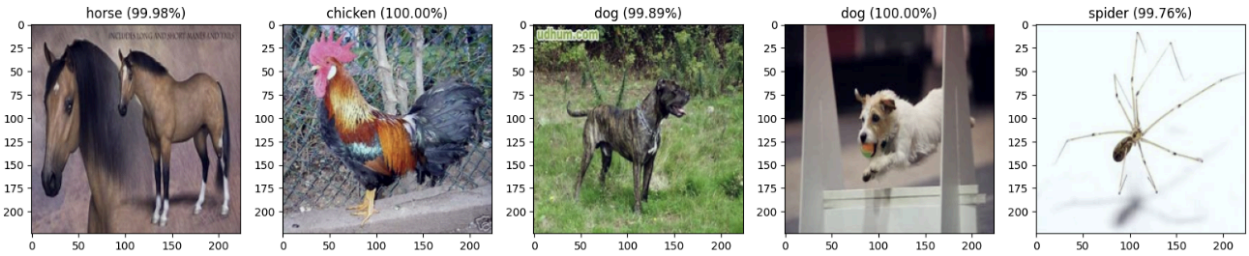


- **Metrics**:
    - **Softmax activation** is applied in the final layer to compute class probabilities.
    - Model performance is assessed using **precision, recall, and F1-score** to measure classification accuracy.
- **Demo Result**:
    - The model's evaluation statistics, including the **confusion matrix**, are displayed.
    - A usage demo illustrates the model's predictions, highlighting cases where it correctly classifies images with high confidence.
    - Incorrect classifications are observed, particularly in cases where confidence scores are low.

Key Functions & Features (Cheat Sheet)

- **Data Processing**:
  - The input images are **downloaded and randomized** to include different orientations.
  - The dataset consists of **2000 images**, divided into:
    - **1400** for training
    - **300** for validation
    - **300** for testing
- **Training & Evaluation**:
  - The training dataset is used to **train and save the model** for future use.
  - Performance is visualized using a **confusion matrix** to analyze class-wise accuracy.
  - Model predictions are displayed, showing the **confidence percentage** for each classification.





3-2. Object Detection with YOLOv8 (Basic Script)

Model Overview

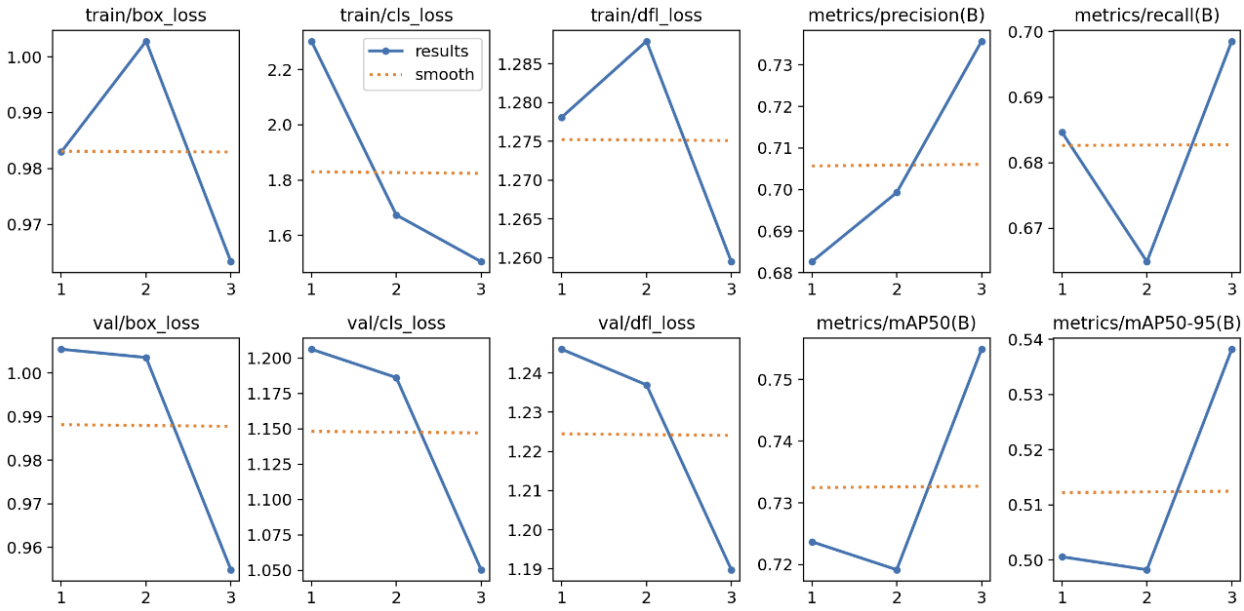
- **Input**: A dataset containing images with **20 object categories**, including **person, bird, cat, dog, horse, sheep, airplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, and TV monitor**.

- **Output**: The model detects and classifies **multiple objects** in an image, assigning each **object class** along with a **confidence score**.

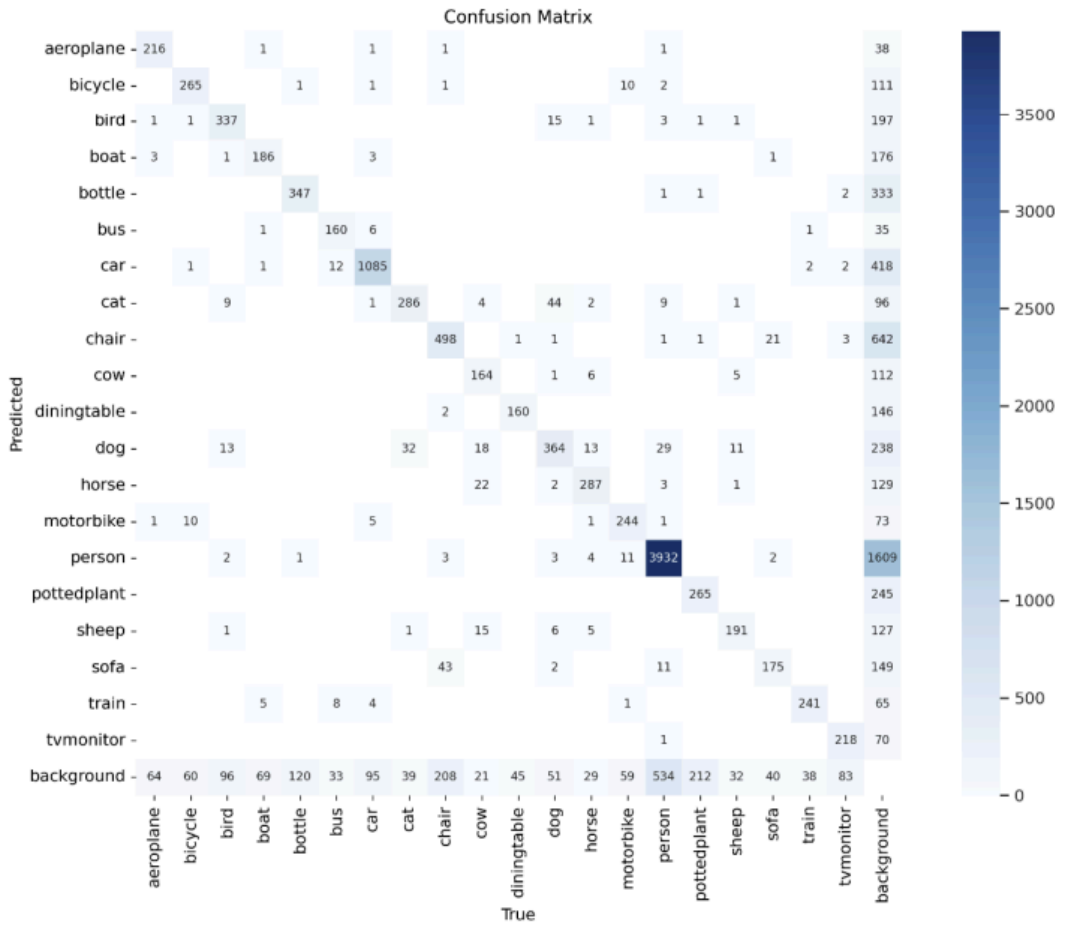
	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.block.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]

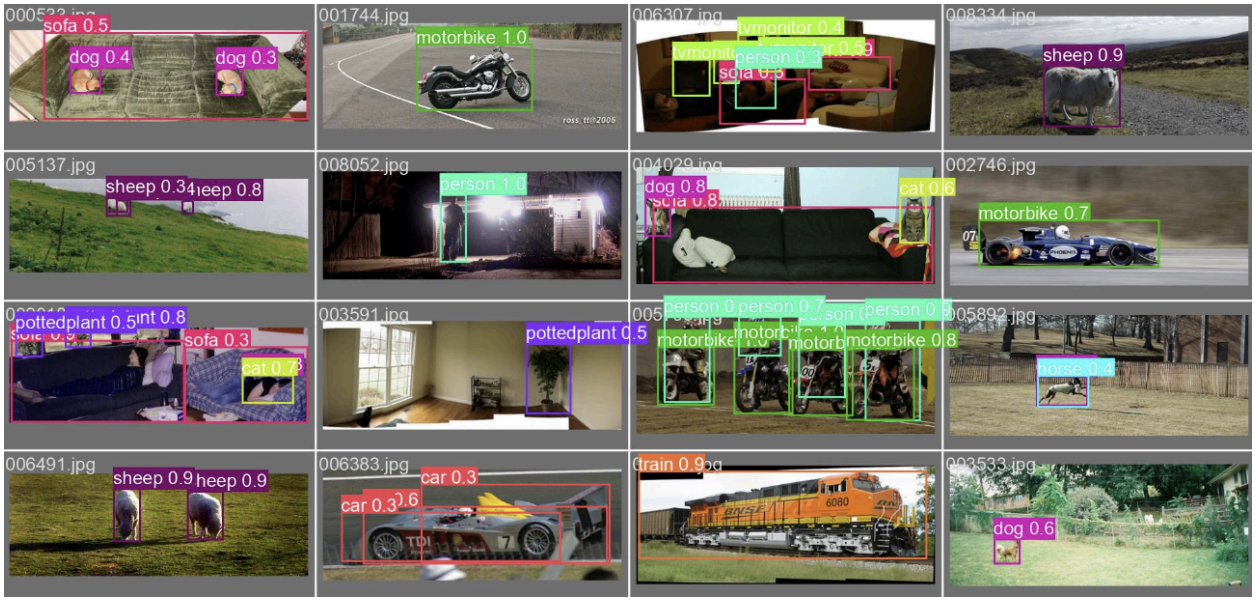
...

- **Hardware Requirements**:
  - The model is designed to run on **CUDA (Nvidia GPU)** for efficient computation.
  - **YOLOv8** is used as the object detection framework.
  - The training process consists of **3 epochs**.
- **Data Statistics**:
  - Each training image is **resized to 640×640 pixels** to standardize the input.
  - The dataset contains **16,551 images**, providing a large sample size for improved model performance.
- **Learning Curve**:
  - The model learns to detect objects by iterating over the dataset during training.



- **Metrics**:
  - The model's performance is validated using the **VOC validation dataset (VOC val)**.
- **Demo Result**:
  - A confusion matrix is generated to visualize the model's performance in classifying detected objects.
  - The model correctly detects and classifies most objects, though some misclassifications occur.


Confusion Matrix
Predicted
True
aeroplane
bicycle
bird
boat
bottle
bus
car
cat
chair
cow
diningtable
dog
horse
motorbike
person
pottedplant
sheep
sofa
train
tvmonitor
background
3500
3000
2500
2000
1500
1000
500
0


sofa 0.5
dog 0.4
dog 0.3
001744.jpg
motorbike 1.0
tvmonitor 0.4
person 0.3
sofa 0.5
sheep 0.9
005137.jpg
sheep 0.8
008052.jpg
person 1.0
dog 0.8
cat 0.6
002746.jpg
motorbike 0.7
pottedplant 0.5
sofa 0.3
cat 0.7
003591.jpg
pottedplant 0.5
motorbike 0.8
horse 0.4
006491.jpg
sheep 0.9
sheep 0.9
006383.jpg
car 0.3
car 0.3
train 0.9
dog 0.6

Key Functions & Features (Cheat Sheet)

- **Object Detection vs. Image Classification**:
  - Unlike the previous two models, which only **classify whole images**, this model can **detect and classify multiple objects within an image**, identifying their locations.
- **Training Process**:
  - **16,551 images** are used for training, significantly improving the model's accuracy.
  - Training images are resized to **640×640 pixels** before being fed into the model.
- **Evaluation**:
  - The confusion matrix illustrates how accurately the model assigns object classes.
  - The VOC validation dataset is used to measure performance.
  -

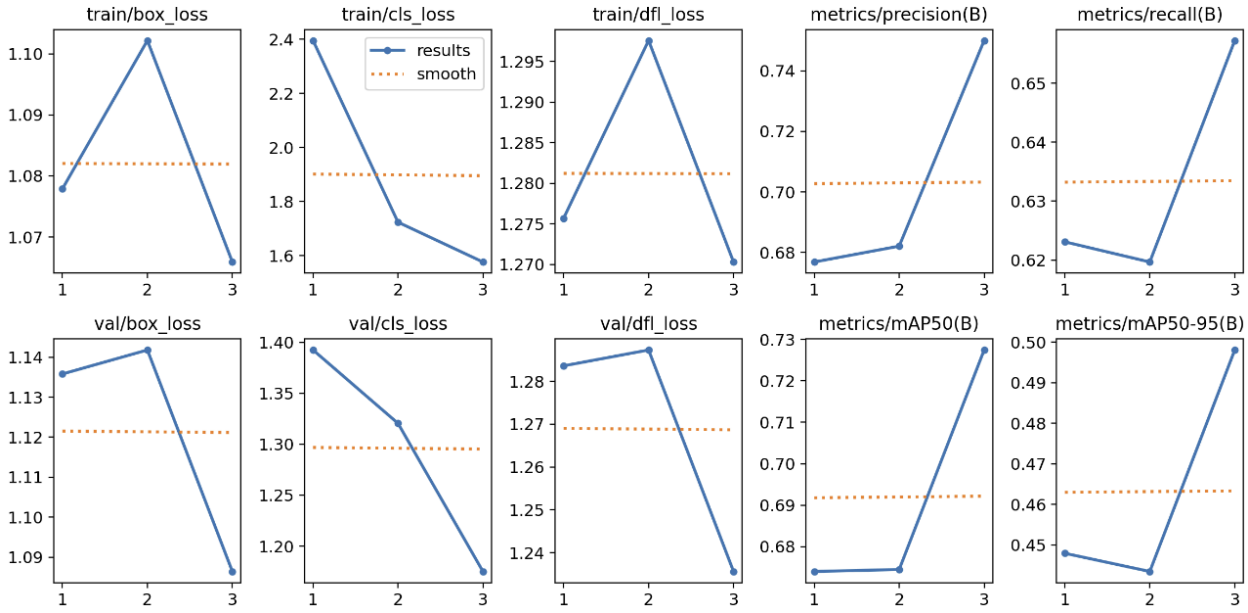
3-3. Object Detection Using YOLOv8 (Custom Dataset)

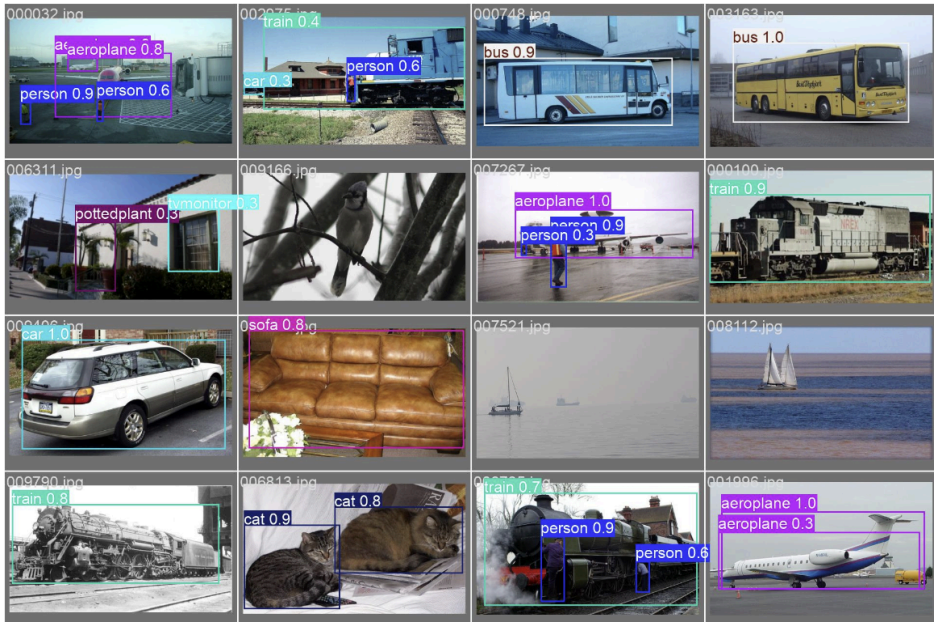
Model Overview

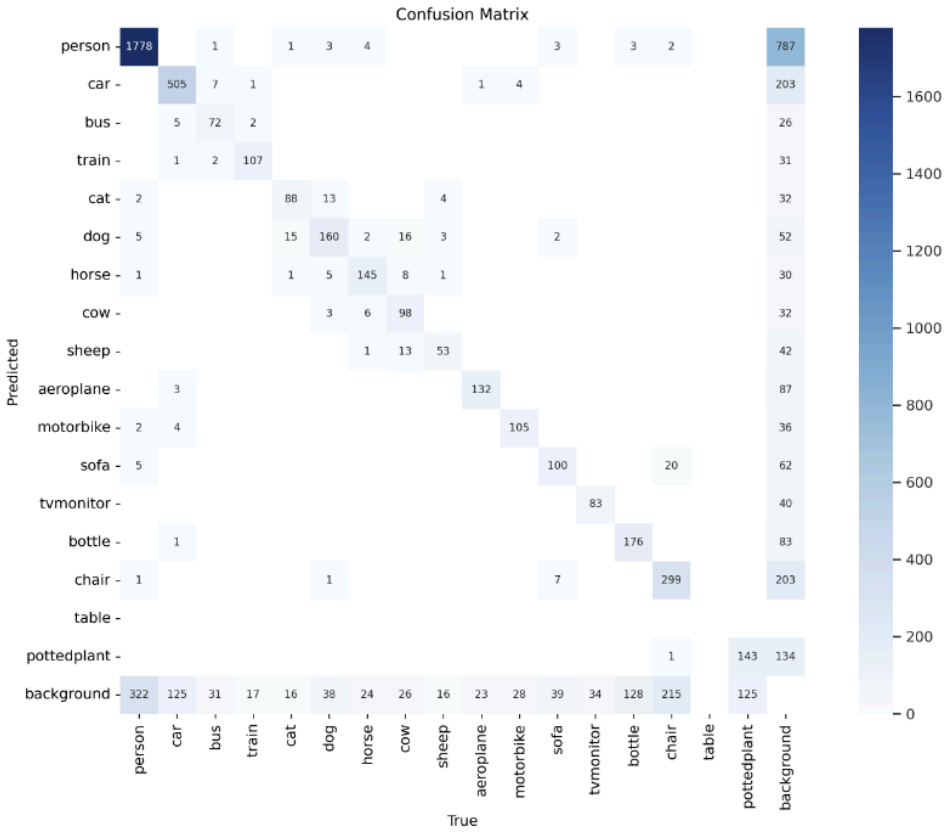
- **Input**: A dataset consisting of images containing **20 object categories**, including **person, bird, cat, dog, horse, sheep, airplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, and TV monitor**.
- **Output**: The model identifies **objects within an image**, assigning them to specific classes along with a **confidence score**.

	from	n	params	module	arguments
0	-1	1	464	ultralytics.nn.modules.conv.Conv	[3, 16, 3, 2]
1	-1	1	4672	ultralytics.nn.modules.conv.Conv	[16, 32, 3, 2]
2	-1	1	7360	ultralytics.nn.modules.block.C2f	[32, 32, 1, True]
3	-1	1	18560	ultralytics.nn.modules.conv.Conv	[32, 64, 3, 2]
4	-1	2	49664	ultralytics.nn.modules.block.C2f	[64, 64, 2, True]
5	-1	1	73984	ultralytics.nn.modules.conv.Conv	[64, 128, 3, 2]
6	-1	2	197632	ultralytics.nn.modules.block.C2f	[128, 128, 2, True]
7	-1	1	295424	ultralytics.nn.modules.conv.Conv	[128, 256, 3, 2]
8	-1	1	460288	ultralytics.nn.modules.block.C2f	[256, 256, 1, True]
9	-1	1	164608	ultralytics.nn.modules.block.SPPF	[256, 256, 5]
10	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
11	[-1, 6]	1	0	ultralytics.nn.modules.conv.Concat	[1]
12	-1	1	148224	ultralytics.nn.modules.block.C2f	[384, 128, 1]
13	-1	1	0	torch.nn.modules.upsampling.Upsample	[None, 2, 'nearest']
14	[-1, 4]	1	0	ultralytics.nn.modules.conv.Concat	[1]
15	-1	1	37248	ultralytics.nn.modules.block.C2f	[192, 64, 1]
16	-1	1	36992	ultralytics.nn.modules.conv.Conv	[64, 64, 3, 2]
17	[-1, 12]	1	0	ultralytics.nn.modules.conv.Concat	[1]
18	-1	1	123648	ultralytics.nn.modules.block.C2f	[192, 128, 1]
19	-1	1	147712	ultralytics.nn.modules.conv.Conv	[128, 128, 3, 2]
20	[-1, 9]	1	0	ultralytics.nn.modules.conv.Concat	[1]

- **Hardware Requirements**:
  - The model runs on **CUDA:0**, utilizing an **Nvidia GPU** for efficient processing.
- **Data Statistics**:
  - The dataset is sourced from a **publicly available dataset**, as referenced in the code.
  - **20% of the dataset** is allocated for testing.
- **Learning Curve & Demo Results**:
  - The model's performance is evaluated through detection accuracy and object classification confidence.







Key Features & Function Overview (Cheat Sheet)

- **Object Detection vs. Image Classification**:
  - Unlike previous models, which solely **classify entire images**, this model can **detect multiple objects within an image** while also classifying them.
- **Public Dataset Utilization**:
  - The model is trained using a **public dataset**, making it adaptable to specific object detection tasks.
  - **20% of the dataset is reserved for testing**, ensuring reliable performance evaluation.
  -

4. Semantic Segmentation Using DeepLabV3 (PyTorch Lightning)

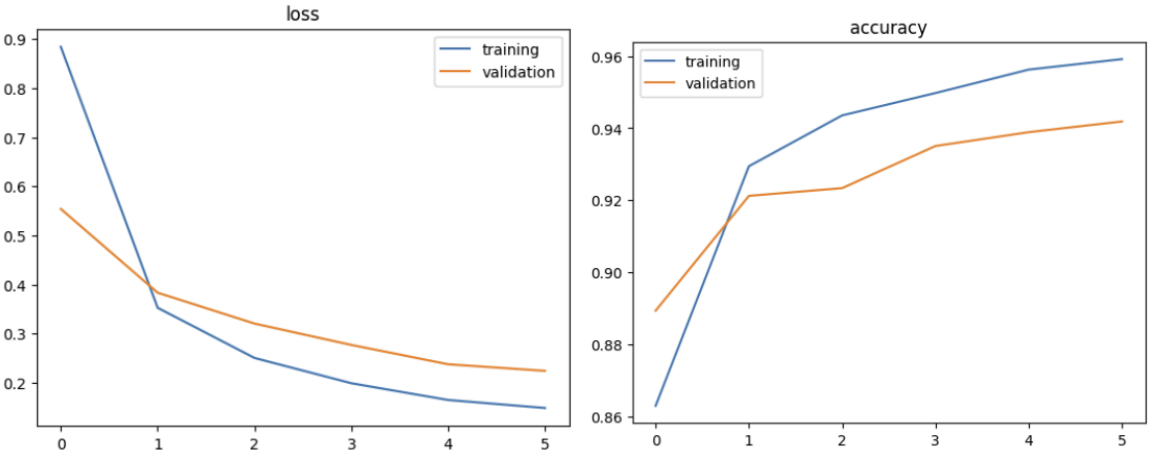
Model Overview

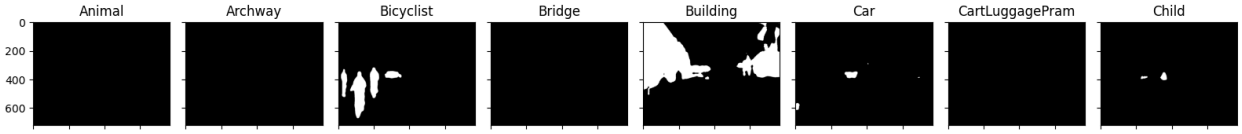
- **Input**:
  - A dataset containing **101 image pairs** with dimensions **960x720 pixels**.
  - Each mask file is labeled with an **"_L" suffix** in the filename.
  - All images and ground truth masks are in **uncompressed 24-bit color PNG format**.

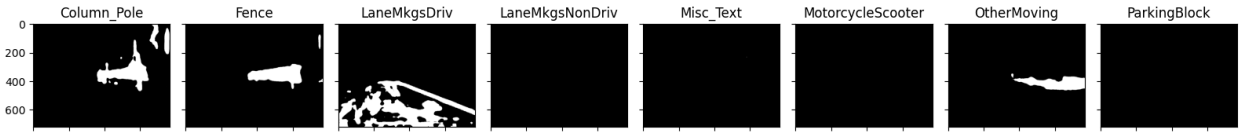
- **Output**:
    - **Segmented images**, where each object in an image is assigned a distinct region.

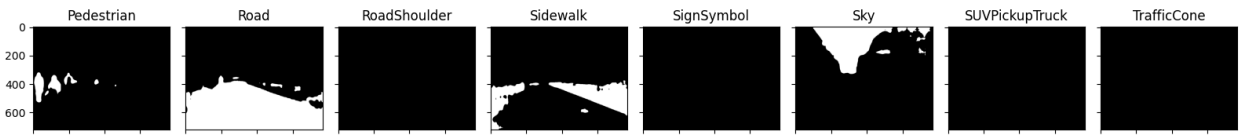
```
DeepLabV3(
  (backbone): IntermediateLayerGetter(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=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)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): Bottleneck(
        (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=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)
...
    (2): ReLU()
    (3): Dropout(p=0.1, inplace=False)
    (4): Conv2d(256, 21, kernel_size=(1, 1), stride=(1, 1))
  )
)
```

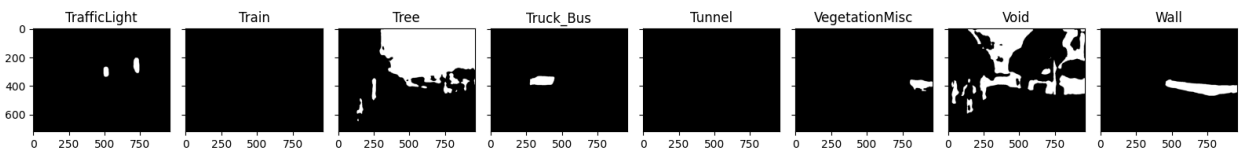
- **Hardware Requirements**:
    - The model runs on **CUDA:0** by default but can fall back to the **CPU** if necessary.
- **Data Statistics**:
    - Dataset split: **70 images for training, 16 for validation, and 15 for testing**.
- **Learning Curve & Demo Results**:
    - The model is assessed by observing its segmentation accuracy and how well it distinguishes different objects within images.

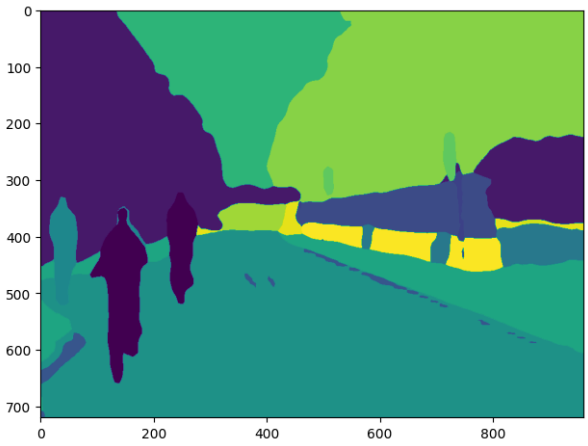

loss
training
validation
accuracy
training
validation


Animal
Archway
Bicyclist
Bridge
Building
Car
CartLuggagePram
Child


Column_Pole
Fence
LaneMkgsDriv
LaneMkgsNonDriv
Misc_Text
MotorcycleScooter
OtherMoving
ParkingBlock


Pedestrian
Road
RoadShoulder
Sidewalk
SignSymbol
Sky
SUVPickupTruck
TrafficCone


TrafficLight
Train
Tree
Truck_Bus
Tunnel
VegetationMisc
Void
Wall



Key Features & Function Overview (Cheat Sheet)

- **DeepLabV3 Model Usage**:
  - The **DeepLabV3** model is trained to **segment images into multiple regions** based on the objects present.
- **Segmentation Output**:
  - The model's output consists of **images segmented into distinct parts**, highlighting the locations of different objects.
- **Performance Evaluation**:
  - The quality of segmentation is assessed visually by comparing the **predicted segmentations** with **ground truth masks**.