



# **CS 4804**

# **Image Classification in**

# **Autonomous Vehicles**

Vatsa Bhatt, Tanisi Tripathi, Fardin Khan, Srikar Iragavarapu



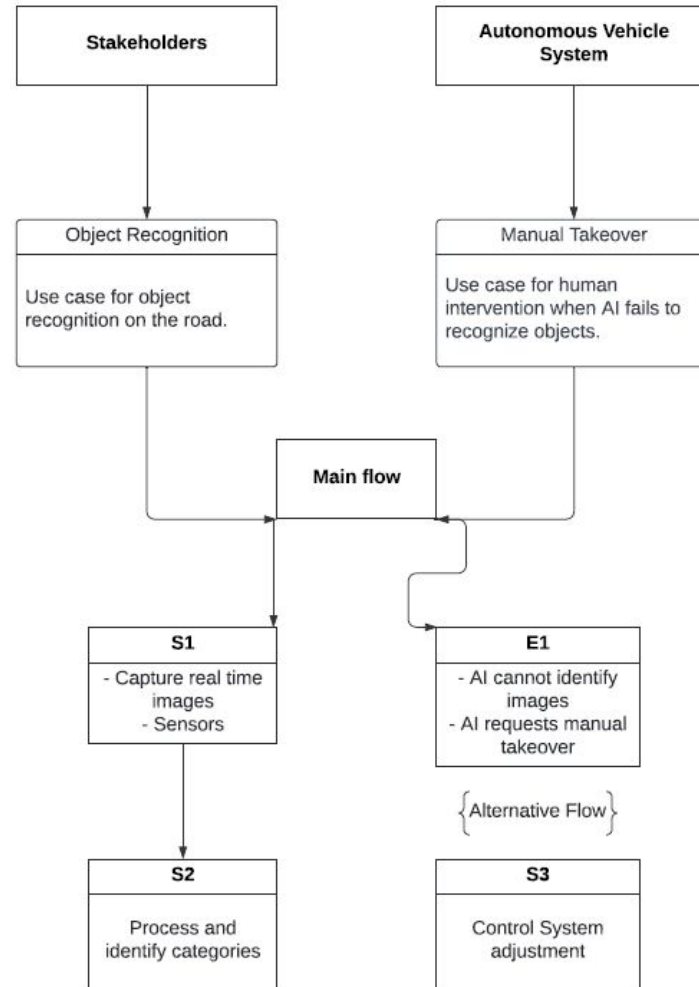
# Problem statement and analysis

- ◆ Address challenges in image classification for vehicles
- ◆ Crucial for ensuring safety
- ◆ Precise identification of road signs, pedestrians, and other vehicles



Fig. 1 Autonomous Vehicles (6)

# Use-Case Scenario



# AI algorithm and model

```
class LargeCNN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(LargeCNN, self).__init__()
        self.input_dim = input_dim
        self.output_dim = output_dim
        self.metrics = {}
        self.flatten = nn.Flatten()
        self.dropout2 = nn.Dropout(0.2)
        self.dropout3 = nn.Dropout(0.3)
        self.relu = nn.ReLU()
        self.maxpool = nn.MaxPool2d(2)
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1)
        self.batchnorm1 = nn.BatchNorm2d(64)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.conv4 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1)
        self.batchnorm2 = nn.BatchNorm2d(256)
        self.conv5 = nn.Conv2d(in_channels=256, out_channels=512, kernel_size=3)
        self.conv6 = nn.Conv2d(in_channels=512, out_channels=1024, kernel_size=3)
        self.batchnorm3 = nn.BatchNorm2d(1024)
        self.l1 = nn.Linear(1024*2*2, 512)
        self.l2 = nn.Linear(512, 128)
        self.batchnorm4 = nn.LayerNorm(128)
        self.l3 = nn.Linear(128, output_dim)

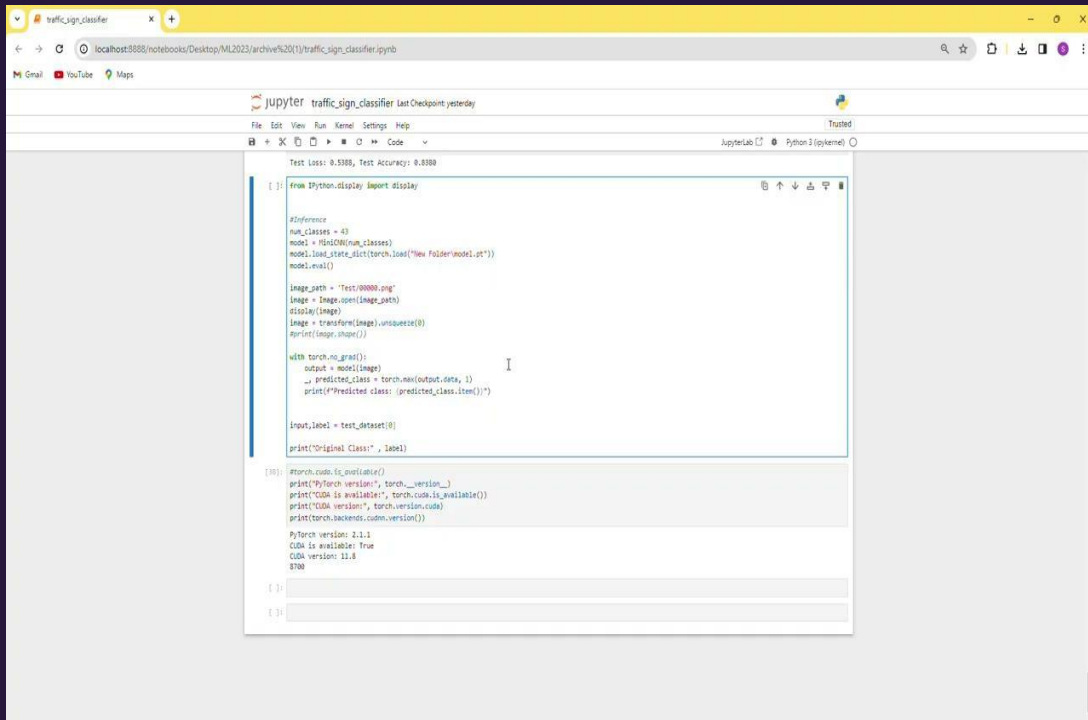
    def forward(self, input):
        conv = self.conv1(input)
        conv = self.conv2(conv)
        batchnorm = self.relu(self.batchnorm1(conv))
        maxpool = self.maxpool(batchnorm)
        conv = self.conv3(maxpool)
        conv = self.conv4(conv)
        batchnorm = self.relu(self.batchnorm2(conv))
        maxpool = self.maxpool(batchnorm)
        conv = self.conv5(maxpool)
        conv = self.conv6(conv)
        batchnorm = self.relu(self.batchnorm3(conv))
        maxpool = self.maxpool(batchnorm)
        flatten = self.flatten(maxpool)
        dense_l1 = self.l1(flatten)
        dropout = self.dropout3(dense_l1)
        dense_l2 = self.l2(dropout)
        batchnorm = self.batchnorm4(dense_l2)
        dropout = self.dropout2(batchnorm)
        output = self.l3(dropout)
        return output
```

The algorithm using a convolutional neural network (CNN) in PyTorch. This algorithm is designed for the German Traffic Sign Recognition Benchmark (GTSRB) dataset.

```
class MiniCNN(nn.Module):
    def __init__(self, num_classes):
        super(MiniCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128 * 4 * 4, 256)
        self.fc2 = nn.Linear(256, num_classes)
        self.dropout = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 128 * 4 * 4)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

# Results and demonstration



```
Test Loss: 0.5388, Test Accuracy: 0.8380

[ ]: from IPython.display import display

#Inference
num_classes = 43
model = HMAOH(num_classes)
model.load_state_dict(torch.load("New Folder\model.pt"))
model.eval()

image_path = "Test\00000.png"
image = Image.open(image_path)
display(image)
image = transform(image).unsqueeze(0)
print(image.shape)

with torch.no_grad():
    output = model(image)
    _, predicted_class = torch.max(output.data, 1)
    print(f"Predicted class: {predicted_class.item()}")

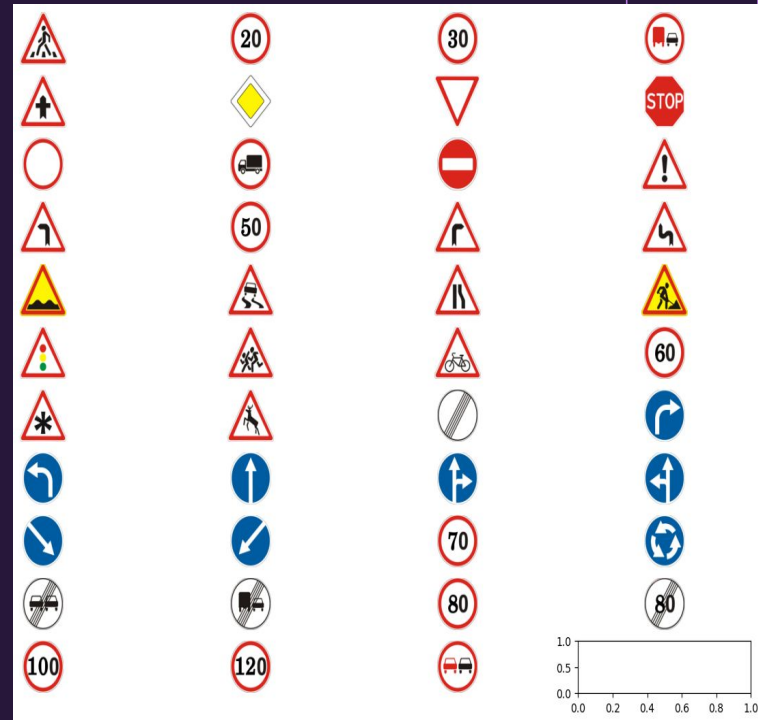
Input label = test_dataset[0]
print("Original Class" , label)

[30]: torch.cuda.is_available()
print("PyTorch version", torch.__version__)
print("CUDA is available", torch.cuda.is_available())
print("CUDA version", torch.version.cuda)
print(torch.backends.cudnn.version())

PyTorch version: 1.1.1
CUDA is available: True
CUDA version: 11.8
8700

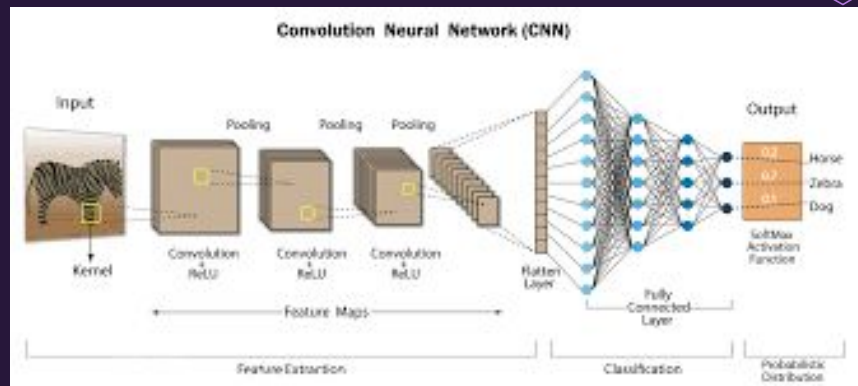
[ ]:

[ ]:
```



# Lessons learned

- ◆ Balancing CNN architecture for efficiency
- ◆ Overcame intricacies in real-world traffic scenarios
- ◆ PyTorch for the flexibility and community support





**Thank you!**

**Questions?**

