CS-300 Artificial Intelligence

FINAL PROJECT

REPORT

2159001 - Võ Quang Dũng
2159005 - Nguyễn Huy Hoàng
2159011 - Nguyễn Ngọc Phú
2159020 - Nguyễn Đỗ Hải Duy

MỤC LỤC

01

Dataset

CIFAR-10

Load CIFAR-10 dataset, create flipped dataset, and split into train and test sets

```python
def prepare_datasets():
    # Define the transformation pipeline for image preprocessing
    transform = transforms.Compose([
        transforms.ToTensor(),  # Convert the image to a PyTorch tensor
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Normalize the image pixel values
    ])

    # Create a flipped version of the CIFAR-10 dataset
    flipped_cifar10 = FlippedCIFAR10(root='./data', train=True, download=True, transform=transform)

    # Determine the sizes for the train and test sets
    train_size = int(0.8 * len(flipped_cifar10))
    test_size = len(flipped_cifar10) - train_size

    # Randomly split the flipped dataset into train and test sets
    train_dataset, test_dataset = random_split(flipped_cifar10, [train_size, test_size])

    # Create DataLoader objects for the train and test sets
    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

    # Print the sizes of the training and testing sets
    print(f"Training set size: {len(train_loader.dataset)}")
    print(f"Testing set size: {len(test_loader.dataset)}")

    return train_loader, test_loader
```

✓ 0.0s

CIFAR-10 class names

```python
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

✓ 0.0s

- **Hàm prepare_datasets() này giúp chuẩn bị dữ liệu huấn luyện và kiểm tra từ tập dữ liệu CIFAR-10 đã lật.**
- **Sử dụng transforms.Compose để tạo một pipeline cho việc tiền xử lý hình ảnh.**
- **Tạo lớp FlippedCIFAR10 để tạo một phiên bản đã lật của tập dữ liệu training của CIFAR-10 (60.000 ảnh) với các tham số như " root='./data' ", " train=True ", " download=True ", " transform=transform "**
- **Sử dụng random_split để ngẫu nhiên phân chia tập dữ liệu đã lật thành tập huấn luyện và tập kiểm tra với tỷ lệ 8:2.**
- **Sử dụng lớp DataLoader từ torch.utils.data để tạo các đối tượng DataLoader cho tập huấn luyện và tập kiểm tra.**
- **Sử dụng lệnh "print" để in ra màn hình kích thước của tập huấn luyện và tập kiểm tra.**
- **"train_loader", "test_loader" là hai đối tượng DataLoader đã được tạo ra.**

02

Kiến trúc mô hình

ResNet18

```
def create_model():
    # Load a pretrained ResNet-18 model
    model = torchvision.models.resnet18(pretrained=True)
    # Replace the fully connected layer with a new one for binary classification
    num_ftrs = model.fc.in_features
    model.fc = nn.Linear(num_ftrs, 2)  # 2 classes: flipped or unflipped
    return model
```

✓ 0.0s

- **Sử dụng 1 phiên bản có sẵn của resnet18 từ torchvision.models cho model đầu tiên với tham số Linear(model.fc.un_features,2). Số 2 ở đây đại diện cho số lớp đầu ra của model, trường hợp thường thấy là nhị phân.**

NeuralNet

```
# Define a function to create a neural network model for binary classification
def create_model():
    # Define a custom neural network class that inherits from nn.Module
    class NeuralNet(nn.Module):
        def __init__(self):
            super(NeuralNet, self).__init__()  # Call the constructor of the superclass
            self.l1 = nn.Linear(3 * 32 * 32, 500)  # Define the first linear layer with input size 3*32*32 and output size 500
            self.relu = nn.ReLU()  # Define the ReLU activation function
            self.l2 = nn.Linear(500, 2)  # Define the second linear layer with input size 500 and output size 2 for binary classification

        def forward(self, x):
            x = x.view(x.size(0), -1)  # Flatten the input tensor to a 1D tensor
            out = self.l1(x)  # Pass the input through the first linear layer
            out = self.relu(out)  # Apply ReLU activation
            out = self.l2(out)  # Pass the output through the second linear layer
            return out  # Return the final output

    # Instantiate the NeuralNet class to create the model
    model = NeuralNet()

    return model  # Return the created model
```

✓ 0.0s

- **NeuralNet là một lớp mạng nơ-ron custom được tạo thủ công bằng cách kế thừa từ module nn trong thư viện của Pytorch, nó cung cấp các lớp và hàm tiện ích cho việc tạo ra các cấu trúc nơ-ron bao gồm nn.Linear, nn.ReLU.**
- **Trong hàm khởi tạo "__init__" có các lớp linear và các hàm kích hoạt bao gồm:**
  - **"self.l1" : lớp linear thứ 1 nhận ảnh màu với kích thước đầu vào là 32*32 và kích thước đầu ra là 500.**
  - **"self.relu" : lớp này kích hoạt ReLU. ReLu cho phép các dự liệu đầu vào lớn hơn 0 được giữ nguyên và nếu chúng nhỏ hơn hoặc bằng sẽ được mặc định thành 0.**
  - **"self.l2" : lớp linear thứ 2 nhận bài toán phân loại nhị phân với kích thước đầu vào là 500 và kích thước đầu ra là 2(nhị phân).**
  - **Hàm Linear có phương trình y=Wx+b, trong đó W là ma trận trọng số, x là vector đầu vào, b là vector bias, và y là vector đầu ra.**
- **Hàm "forward" cho phép xác định cách dữ liệu truyền qua mạng nơ-ron.**
  - **x.view(x.size(0), -1): Biến đổi tensor đầu vào x thành một tensor 1 chiều. Kích thước của batch được giữ nguyên (x.size(0)), còn chiều dài của vector (feature vector) được tính tự động bằng cách để -1.**
  - **Sau đó dữ liệu được truyền vào một mạng nơ-ron tương tự như được giải thích bên trên.**

03

Kết quả đạt được

Model training:

Resnet18 - Epochs: 10

```
Epoch 1, Loss: 0.4434, Accuracy: 78.61%
Epoch 2, Loss: 0.3372, Accuracy: 84.89%
Epoch 3, Loss: 0.2928, Accuracy: 87.13%
Epoch 4, Loss: 0.2645, Accuracy: 88.39%
Epoch 5, Loss: 0.2345, Accuracy: 90.13%
Epoch 6, Loss: 0.2199, Accuracy: 90.86%
Epoch 7, Loss: 0.1949, Accuracy: 91.93%
Epoch 8, Loss: 0.1824, Accuracy: 92.50%
Epoch 9, Loss: 0.1580, Accuracy: 93.57%
Epoch 10, Loss: 0.1834, Accuracy: 92.58%
Training complete in 15.0m 41.2132682800293s
```

```
Finished Training
Accuracy of class 0: 85.33%
Accuracy of class 1: 96.46%
Accuracy of class 2: 85.98%
Accuracy of class 3: 74.53%
Accuracy of class 4: 87.41%
Accuracy of class 5: 85.25%
Accuracy of class 6: 85.45%
Accuracy of class 7: 95.91%
Accuracy of class 8: 94.82%
Accuracy of class 9: 95.63%
```

```
Top 3 classes with lowest accuracy:
Class 3: 74.53%
Class 5: 85.25%
Class 0: 85.33%
```

Chú thích :

1. **Epoch: Vòng lặp huấn luyện**
2. **Accuracy of Class: Độ chính xác của lớp**
3. **Loss: Mức độ sai lệch**

03

Kết quả đạt được

Model training:

NeuralNet - Epochs: 10

```
Epoch 1, Loss: 0.5665, Accuracy: 70.11%
Epoch 2, Loss: 0.5436, Accuracy: 72.06%
Epoch 3, Loss: 0.5342, Accuracy: 72.79%
Epoch 4, Loss: 0.5211, Accuracy: 73.67%
Epoch 5, Loss: 0.5161, Accuracy: 74.13%
Epoch 6, Loss: 0.4728, Accuracy: 76.93%
Epoch 7, Loss: 0.4617, Accuracy: 77.67%
Epoch 8, Loss: 0.4548, Accuracy: 78.00%
Epoch 9, Loss: 0.4497, Accuracy: 78.28%
Epoch 10, Loss: 0.4440, Accuracy: 78.64%
Training complete in 2.0m 45.068302392959595s
```

```
Accuracy of class 0: 76.93%
Accuracy of class 1: 83.68%
Accuracy of class 2: 69.57%
Accuracy of class 3: 57.10%
Accuracy of class 4: 75.25%
Accuracy of class 5: 61.89%
Accuracy of class 6: 69.04%
Accuracy of class 7: 85.01%
Accuracy of class 8: 87.91%
Accuracy of class 9: 89.53%
```

```
Top 3 classes with lowest accuracy:
Class 3: 57.10%
Class 5: 61.89%
Class 6: 69.04%
```

Chú thích :

1. **Epoch: Vòng lặp huấn luyện**
2. **Accuracy of Class: Độ chính xác của lớp**
3. **Loss: Mức độ sai lệch**

03

Kết quả đạt được

So sánh hiệu suất giữa 2 model

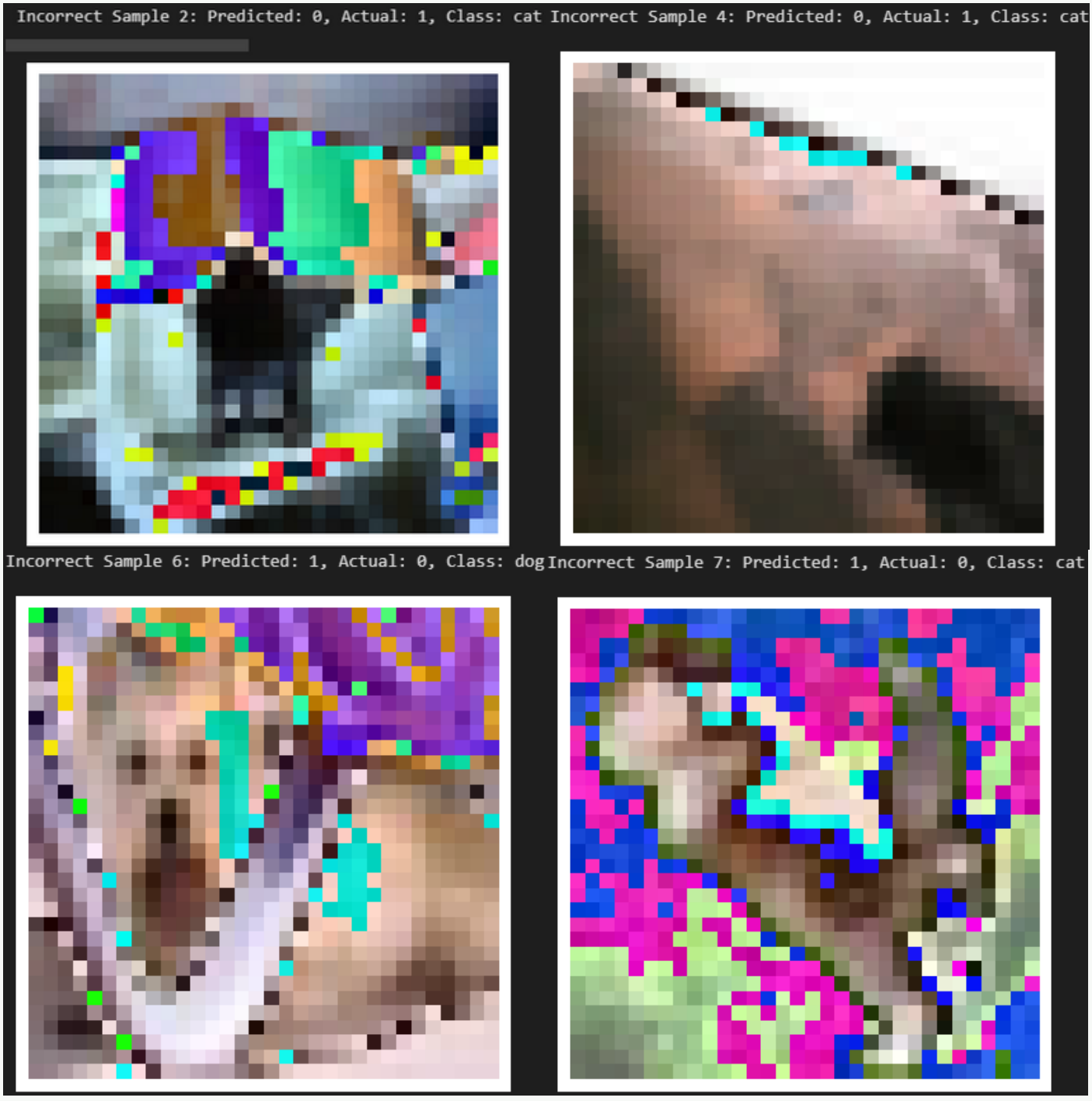
Có thể dễ dàng nhận thấy custom model NeuralNet có kích thước, cấu trúc, số lượng lớp nhỏ hơn và không thể sâu bằng model ResNet18 được pretrained trong pytorch dẫn đến hiệu suất, độ chính xác trong bài toán phân loại hình ảnh cũng không thể bằng ResNet18.

Từ kết quả chạy thử bên trên, ta có thể xác định trong cùng 1 điều kiện (10 epcho) ResNet18 tuy có thời gian chạy lâu hơn nhưng bù lại nó có loss trung bình thấp hơn so NeuralNet. Đồng thời ResNet18 chỉ mất 2 epcho để đạt được mức 80% acc trong khi ResNet18 tuy chạy đủ 10 epcho nhưng vẫn không thể đạt được 80% acc.

Tuy custom model trên thua nhiều về mặt tính năng so với các pretrained model nói chung và ResNet18 nói riêng nhưng nó lại dễ dàng được tạo ra và chỉnh sửa tùy theo yêu cầu của bài toán.

04 Các trường hợp ảnh bị phân loại sai

Model training: Resnet18



04 Các trường hợp ảnh bị phân loại sai

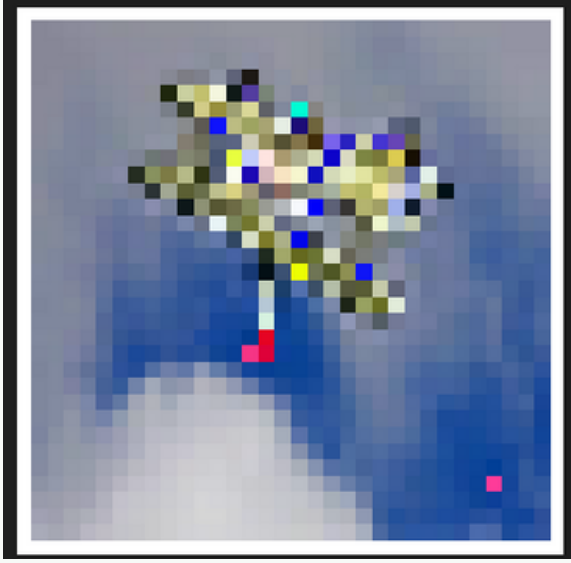
Incorrect Sample 10: Predicted: 0, Actual: 1, Class: dog

Incorrect Sample 11: Predicted: 0, Actual: 1, Class: dog

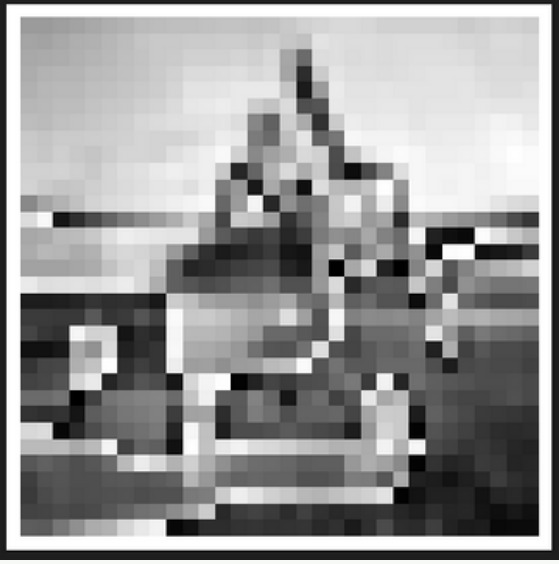
Incorrect Sample 12: Predicted: 1, Actual: 0, Class: dog

Incorrect Sample 16: Predicted: 1, Actual: 0, Class: dog

Incorrect Sample 17: Predicted: 1, Actual: 0, Class: plane

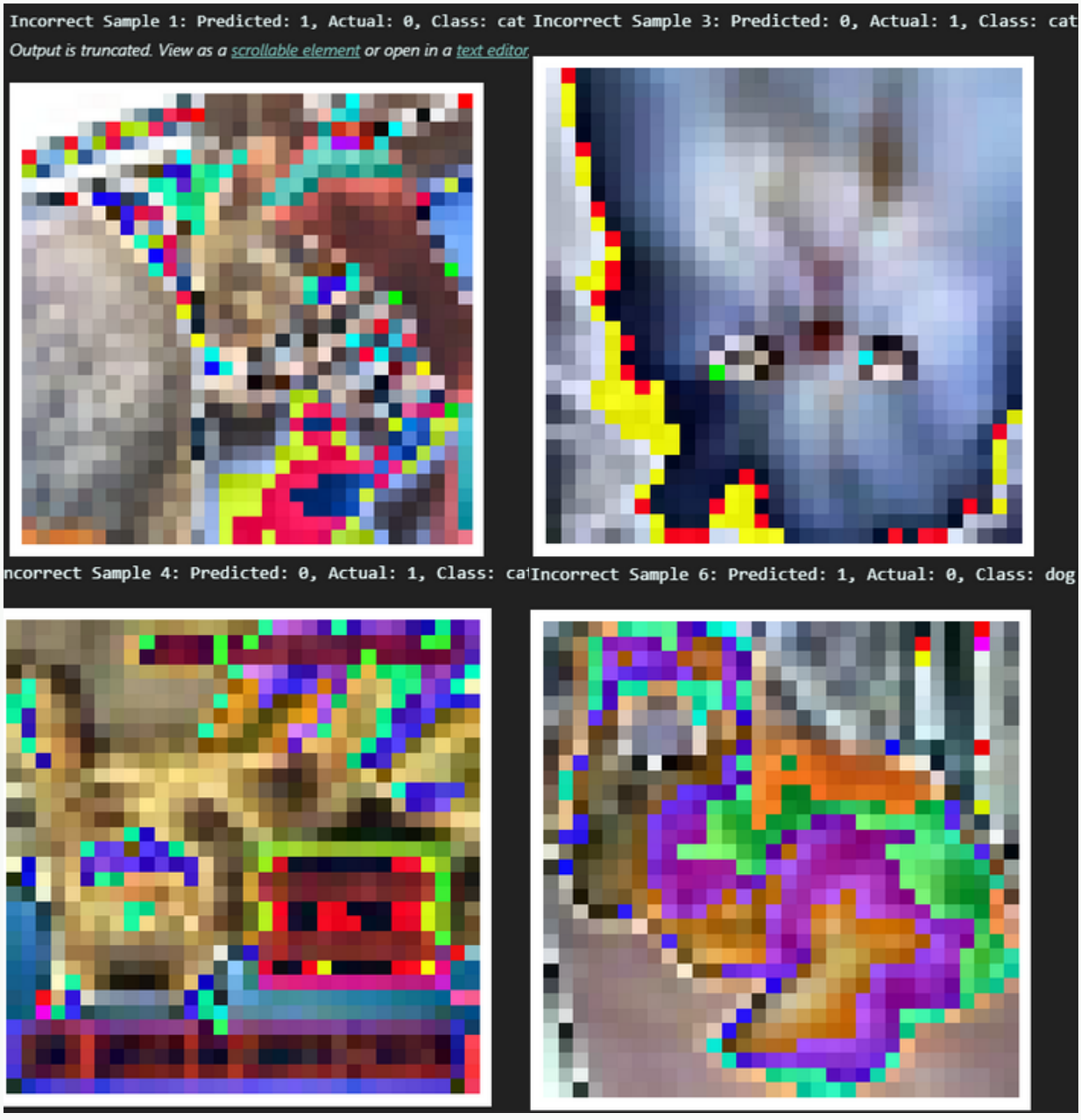


Incorrect Sample 18: Predicted: 1, Actual: 0, Class: plane

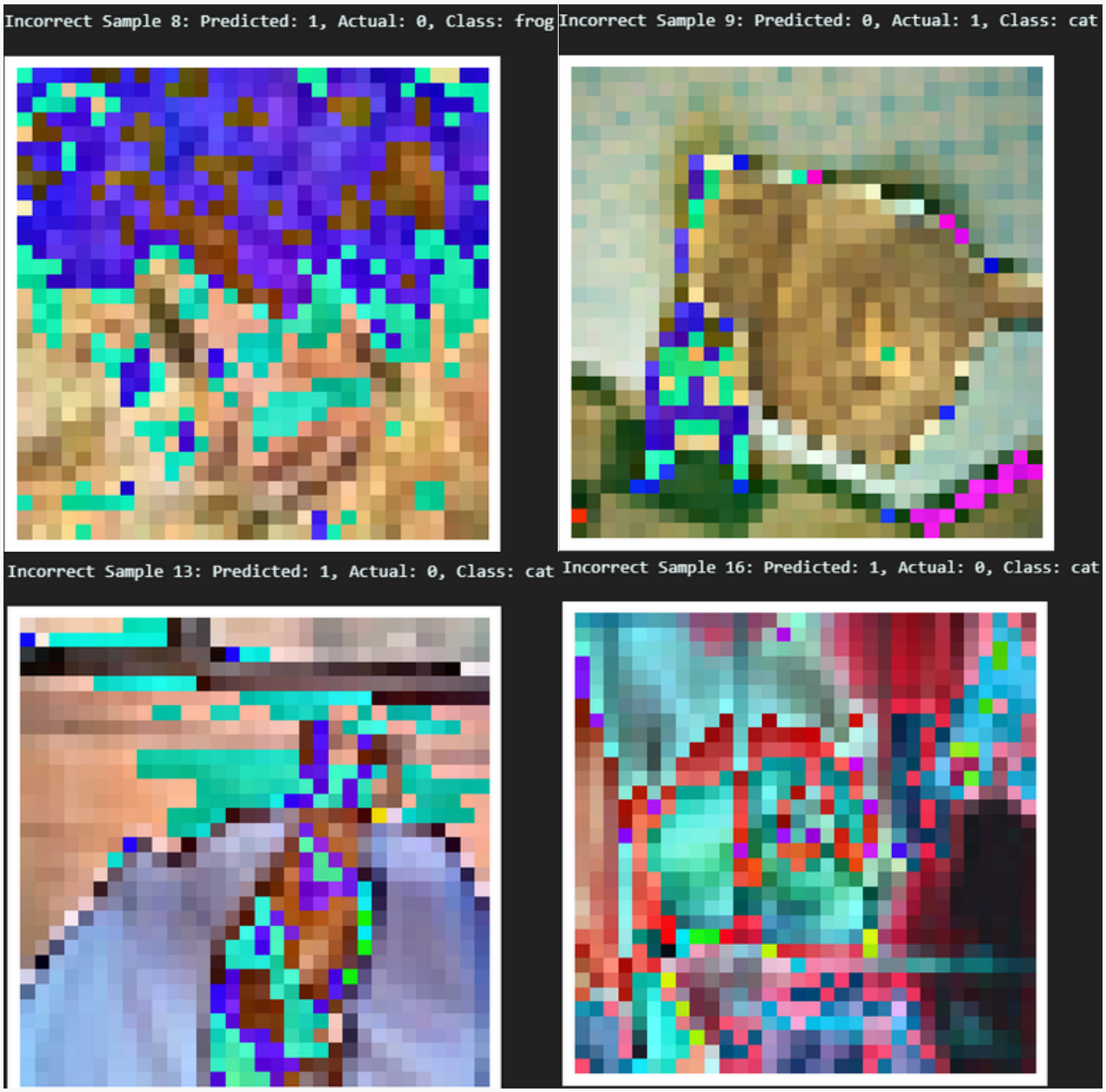


04 Các trường hợp ảnh bị phân loại sai

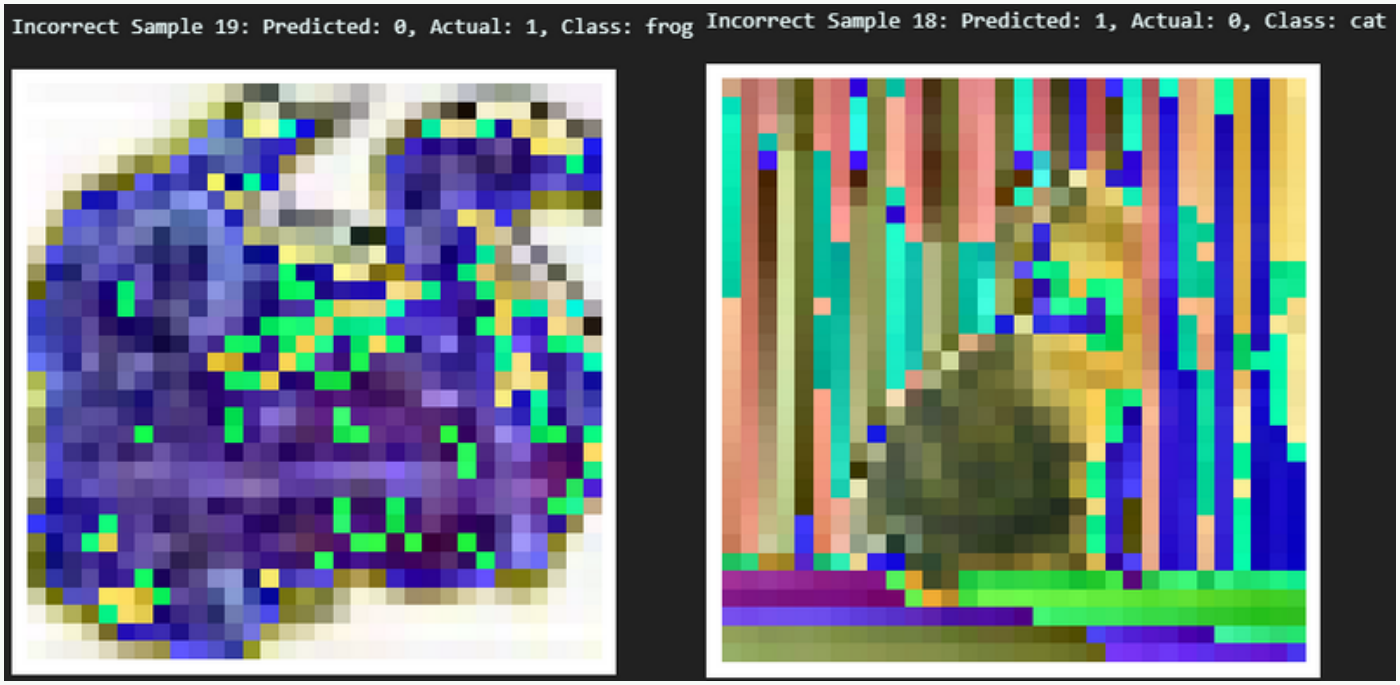
Model training: NeuralNet



04 Các trường hợp ảnh bị phân loại sai



04 Các trường hợp ảnh bị phân loại sai



Nguyên do:

- *Dữ liệu huấn luyện không đủ*: Nếu mô hình không được huấn luyện trên đủ lượng dữ liệu đa dạng, nó có thể không thể nhận biết chính xác các đối tượng trong một số hình ảnh.
- *Độ phức tạp của mô hình*: Mô hình học máy có thể không đủ phức tạp để nắm bắt được tất cả các đặc trưng cần thiết từ hình ảnh.
- *Thiếu điều chỉnh*: Nếu mô hình quá khớp với dữ liệu huấn luyện, nó có thể không thể tổng quát hóa tốt đối với dữ liệu mới.
- *Mô hình không phù hợp*: Mô hình có thể không phù hợp với tác vụ phân loại hình ảnh cụ thể này. Có thể cần một mô hình phức tạp hơn hoặc một mô hình được huấn luyện với một tập dữ liệu khác.