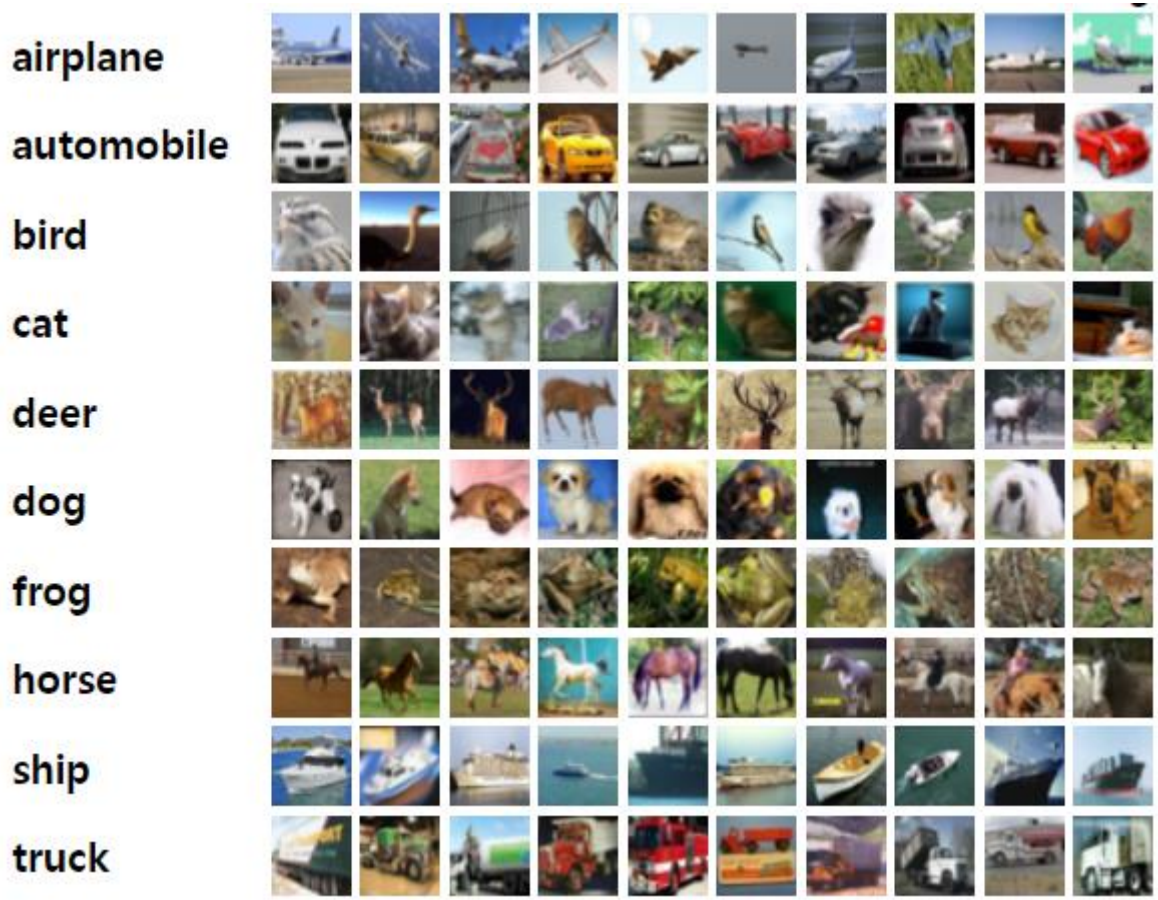
Exercise 5

Image Classification

- **Cifar 10 dataset**



https://www.cs.toronto.edu/~kriz/cifar.html

Image Classification

- Cifar 10 dataset

Classifier

airplane

automobile

bird

cat

deer

dog

frog

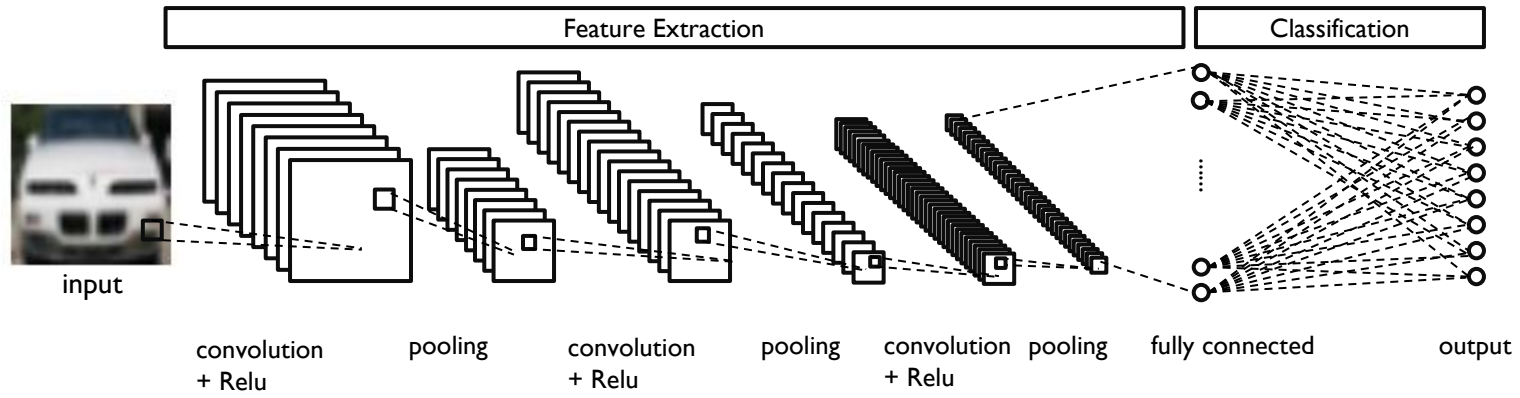
horse

ship

truck

Coding 전에 생각해 볼 것

- 입력 데이터 : image(C , H , W)
- 출력 데이터 : class number
- Optimizer: ?
- Loss function: cross entropy



준비 단계 1

입, 출력 데이터

```
transform_CIFAR10 = transforms.Compose([
    transforms.ToTensor()
])
train_loader = torch.utils.data.DataLoader(
    datasets.CIFAR10(
        root      = './data_CIFAR10',
        train     = True,
        download  = True,
        transform = transform_CIFAR10),
    batch_size=BATCH_SIZE,
    shuffle=True
)
test_loader = torch.utils.data.DataLoader(
    datasets.CIFAR10(
        root      = './data_CIFAR10',
        train     = False,
        download  = True,
        transform = transform_CIFAR10),
    batch_size=BATCH_SIZE,
    shuffle=True
)
```

Model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 8, 30, 30]             224
            Conv2d-2           [-1, 16, 28, 28]           1,168
            Conv2d-3           [-1, 24, 26, 26]           3,480
         Dropout2d-4           [-1, 24, 26, 26]               0
            Linear-5                 [-1, 128]         519,296
            Linear-6                  [-1, 10]           1,290
================================================================
Total params: 525,458
Trainable params: 525,458
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.40
Params size (MB): 2.00
Estimated Total Size (MB): 2.42
----------------------------------------------------------------
```

training and validation

```
def train(model, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(DEVICE), target.to(DEVICE)
        optimizer.zero_grad()
        output = model(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()

        if batch_idx % 200 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
```

```
def evaluate(model, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(DEVICE), target.to(DEVICE)
            output = model(data)
            test_loss += F.cross_entropy(output, target,
                                         reduction='sum').item()
            pred = output.max(1, keepdim=True)[1]
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    test_accuracy = 100. * correct / len(test_loader.dataset)
    return test_loss, test_accuracy
```

```
for epoch in range(1, EPOCHS + 1):
    train(model, train_loader, optimizer, epoch)
    test_loss, test_accuracy = evaluate(model, test_loader)

    print('[{}] Test Loss: {:.4f}, Accuracy: {:.2f}%'.format(
          epoch, test_loss, test_accuracy))
```

Question and Answer