Includes and network parameters

In [1]:

```
import time
import shutil
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import matplotlib.pyplot as plt
import sys
from matplotlib.pyplot import figure

if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

In [2]:

```
# Network parameters
'''
num_neurons : TYPE list
            DESCRIPTION. list of neurons in each layer.
                This should have a minimum length of 3.
                First element represents the dimension of input vector.
                Last element represents the dimension of the output vector.
                middle elements represent the number of neurons in hidden layers
activations : TYPE, option list where each element can be either 'relu' or 'sigm
            DESCRIPTION. The default is ['relu'].
            If len(activations)==1:
                same activation function is applied across all hidden layers.
            else:
                len(activations) should be equal to the number of hidden layers.
'''
num_neurons = [2,20,10,10,2] # list of neurons in each layer of NN.
activations = ['relu'] # represents the activation function used at the hidden l

# optimizer parameters
lr = 0.01
lr_step = [500]
weight_decay = 1e-3

#  training parameters
num_epochs = 200
batch_size = 256

#
print_freq = 10

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

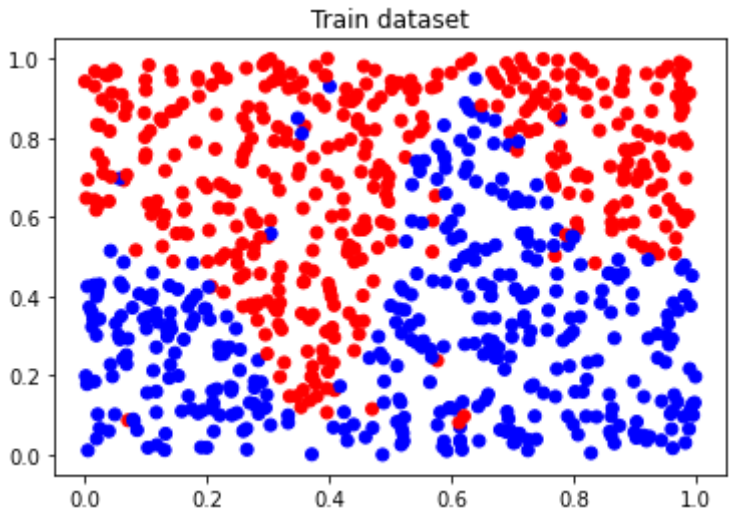
Create and plot data set

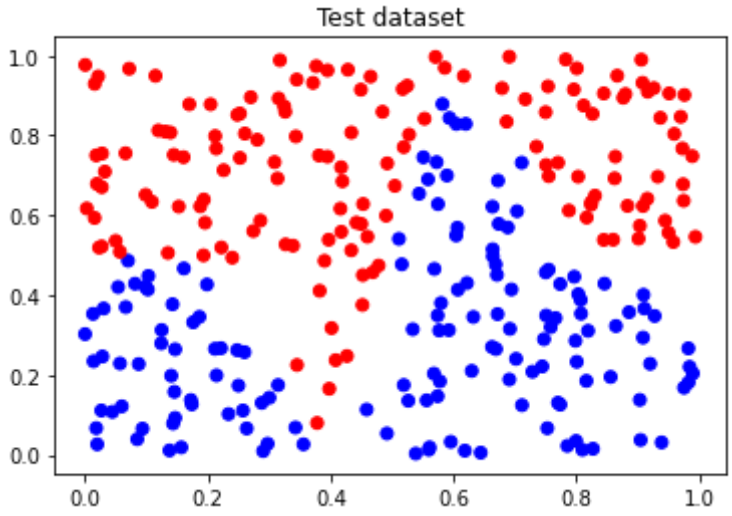
Do not change this cell!

In [3]:

```
# DO NOT change this cell.
ns = 800
np.random.seed(0)
```

```
X_train = np.random.rand(ns,2)
x1 = X_train[:,0]
x2 = X_train[:,1]
y_train = ((np.exp(-((x1-0.5)*6)**2)*2*((x1-0.5)*6)+1)/2-x2)>0

idx = np.random.choice(range(ns),size=(int(ns*0.03),))
y_train[idx] = -y_train[idx]

ns = 300
np.random.seed(1)
X_val = np.random.rand(ns,2)
x1 = X_val[:,0]
x2 = X_val[:,1]
y_val = ((np.exp(-((x1-0.5)*6)**2)*2*((x1-0.5)*6)+1)/2-x2)>0

def plot(X,y,title="Dataset"):
    colors = np.where(y==0, 'r', 'b')
    plt.figure()
    plt.scatter(X[:,0],X[:,1],color=colors)
    plt.title(title)
    plt.show()

plot(X_train,y_train,"Train dataset")
plot(X_val,y_val,"Test dataset")
```





Load data set into Torch dataloader

In [4]:

```
X_train_tensor = torch.Tensor(X_train) # transform to torch tensor
y_train_tensor = torch.Tensor(y_train)

train_dataset = TensorDataset(X_train_tensor,y_train_tensor) # create your datse
train_loader = DataLoader(train_dataset,batch_size=batch_size,shuffle=True,drop_

X_val_tensor = torch.Tensor(X_val) # transform to torch tensor
y_val_tensor = torch.Tensor(y_val)

val_dataset = TensorDataset(X_val_tensor,y_val_tensor) # create your datset
val_loader = DataLoader(val_dataset,batch_size=batch_size,shuffle=True,drop_last
```

Model: Feedforward neural network

In [5]:

```
class LinearNN(nn.Module):

    def __init__(self,num_neurons,activations=['relu']):
        '''
        Parameters
        ----------
        num_neurons : TYPE list
            DESCRIPTION. list of neurons in each layer.
                This should have a minimum length of 3.
                First element represents the dimension of input vector.
                Last element represents the dimension of the output vector.
                middle elements represent the number of neurons in hidden layers

        activations : TYPE, optional list.
            DESCRIPTION. The default is ['relu'].
            If len(activations)==1:
                same activation function is applied across all hidden layers.
            else:
                len(actiavtions) should be equal to the number of hidden layers.

        Returns
        -------
        None.

        '''

        super(LinearNN,self).__init__()
        assert isinstance(num_neurons,list)
        assert np.all([isinstance(neurons,int) for neurons in num_neurons])
        assert np.all([neurons>=1 for neurons in num_neurons])
        assert len(num_neurons)>=3
        if activations is not None:
            assert isinstance(activations,(list))
            assert (len(activations)==len(num_neurons)-2) or (len(activations)==


        def activation_layer(act_func):
            '''


            Parameters
            ----------
            act_func : TYPE should be one from {'relu','sigmoid','tanh'}.
                DESCRIPTION.
```

```
            Raises
            ------
            NotImplementedError
                DESCRIPTION.

            Returns
            -------
            TYPE
                DESCRIPTION.

            '''
            if act_func=='relu':
                return nn.ReLU(inplace=True)
            elif act_func=='sigmoid':
                return nn.Sigmoid()
            elif act_func=='tanh':
                return nn.Tanh()
            else:
                raise NotImplementedError


        layers = []
        for idx,_ in enumerate(num_neurons[:-1]):
            layers.append(nn.Linear(in_features=num_neurons[idx],
                                    out_features=num_neurons[idx+1],
                                    bias=True))

            if idx!=len(num_neurons)-2: # add activation for all layers except t
                if len(activations)==1:
                    layers.append(activation_layer(activations[0]))
                else:
                    layers.append(activation_layer(activations[idx]))

        self.network = nn.Sequential(*layers)


    def forward(self,x):
        x = self.network(x)
        return x


def linear_nn(num_neurons,activations=['relu']):
    model = LinearNN(num_neurons,activations)
    return model
```

Define training function

In [6]:

```
def train(train_loader, model, criterion, optimizer, epoch):
    batch_time = AverageMeter()
    data_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()

    # switch to train mode
    model.train()

    end = time.time()
    for i, (input, target) in enumerate(train_loader):
```

```
        # measure data loading time
        data_time.update(time.time() - end)

        target = target.to(device)
        input_var = torch.autograd.Variable(input).to(device)
        target_var = torch.autograd.Variable(target).to(device)
        # target_var = torch.squeeze(target_var)
        # compute output
        output = model(input_var)

        # compute loss
        loss = criterion(output, target_var.long())

        # measure accuracy and record loss
        prec1 = accuracy(output.data, target)
        losses.update(loss.item(), input.size(0))
        top1.update(prec1[0][0], input.size(0))

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        if i % print_freq == 0:
            curr_lr = optimizer.param_groups[0]['lr']
            print('Epoch: [{0}/{1}][{2}/{3}]\t'
                  'LR: {4}\t'
                  'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                  'Train Acc {top1.val:.3f} ({top1.avg:.3f})'.format(
                   epoch, num_epochs, i, len(train_loader), curr_lr,
                   loss=losses, top1=top1))


    print(' * Train Acc {top1.avg:.3f}'.format(top1=top1))
```

Define validation and prediction functions

In [7]:

```
def validate(val_loader, model, criterion):
    batch_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()

    # switch to evaluate mode
    model.eval()

    end = time.time()
    for i, (input, target) in enumerate(val_loader):
        target = target.to(device)
        input_var = torch.autograd.Variable(input, volatile=True).to(device)
        target_var = torch.autograd.Variable(target, volatile=True).to(device)

        # compute output
        output = model(input_var)
        # loss = criterion(output, target_var[:,None])
```

```
        loss = criterion(output, target_var.long())

        # measure accuracy and record loss
        prec1 = accuracy(output.data, target)
        losses.update(loss.item(), input.size(0))
        top1.update(prec1[0][0], input.size(0))

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

        if i % print_freq == 0:
            print('Test: [{0}/{1}]\t'
                  'Loss {loss.val:.4f} ({loss.avg:.4f})\t'
                  'Prec@1 {top1.val:.3f} ({top1.avg:.3f})'.format(
                   i, len(val_loader), loss=losses,
                   top1=top1))


    print(' * Test Acc {top1.avg:.3f}'.format(top1=top1))

    return top1.avg

def predict(dataloader,model):
    y_pred = []
    y_true = []
    x = []
    with torch.no_grad():
        for i, (input, target) in enumerate(dataloader):
            # target = target.to(device)
            input_var = torch.autograd.Variable(input, volatile=True).to(device)
            # target_var = torch.autograd.Variable(target, volatile=True).to(dev

            # compute output
            output = model(input_var)
            labels = torch.argmax(output,axis=1)
            y_pred.extend(list(labels.data.detach().cpu().numpy()))
            y_true.extend(list(target.numpy()))
            x.extend(list(input_var.data.detach().cpu().numpy()))
    return np.array(x),np.array(y_true),np.array(y_pred)
```

Function to plot the decision boundary of the neural network

In [8]:

```
def plot_decision_boundary(model):
    h = 0.005
    x_min, x_max = 0,1
    y_min, y_max = 0,1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    x1 = xx.ravel()
    x2 = yy.ravel()
    y = ((np.exp(-((x1-0.5)*6)**2)*2*((x1-0.5)*6)+1)/2-x2)>0

    X_train_tensor = torch.Tensor(np.c_[xx.ravel(), yy.ravel()]) # transform to
    y_train_tensor = torch.Tensor(y)

    dataset = TensorDataset(X_train_tensor,y_train_tensor) # create your datset
    dataloader = DataLoader(dataset,batch_size=batch_size,shuffle=False,drop_las
```

```
    x,y_true,y_pred = predict(dataloader,model)
    Z = y_pred.reshape(xx.shape)
    plt.figure()
    plt.contourf(x[:,0].reshape(xx.shape), x[:,1].reshape(xx.shape), Z, cmap=plt
    plt.axis('tight')

    # scatter plot of data points with colors corresponding to the correct label
    ns = 500
    np.random.seed(0)
    X_test = np.random.rand(ns,2)
    x1 = X_test[:,0]
    x2 = X_test[:,1]
    y_test = ((np.exp(-((x1-0.5)*6)**2)*2*((x1-0.5)*6)+1)/2-x2)>0
    colors = np.where(y_test==0, 'r', 'b')
    plt.scatter(x1,x2,color=colors)
    # plt.scatter(x[:,0],x[:,1],colors=)
    plt.show()
```

Functions to track the model performance and save the desired model state

In [9]:

```
def save_checkpoint(state, is_best, filename='checkpoint.pth.tar'):
    torch.save(state, filename)
    if is_best:
        shutil.copyfile(filename, 'model_best.pth.tar')


class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count


def accuracy(output, target, topk=(1,)):
    """Computes the precision@k for the specified values of k"""
    maxk = max(topk)
    batch_size = target.size(0)

    _, pred = output.topk(maxk, 1, True, True)
    pred = pred.t()
    correct = pred.eq(target.view(1, -1).expand_as(pred))

    res = []
    for k in topk:
        correct_k = correct[:k].view(-1).float().sum(0, keepdim=True)
```

```
        res.append(correct_k.mul_(100.0 / batch_size))
    return res
```

Create model instance; define loss function and optimizer

In [10]:

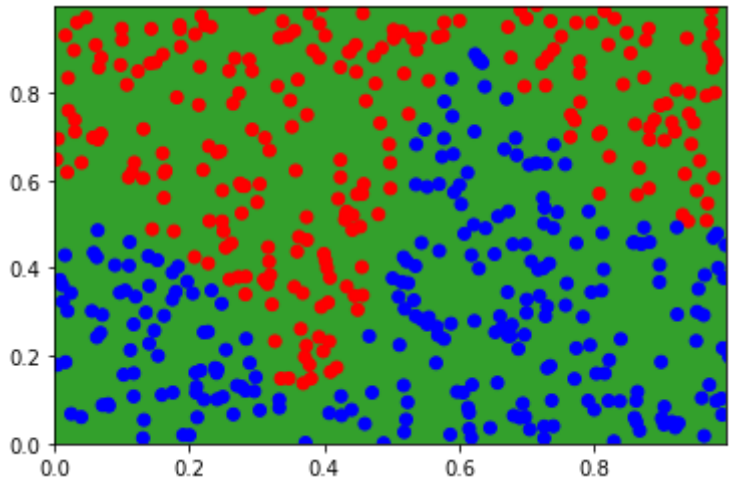
```
torch.manual_seed(999)
model = linear_nn(num_neurons,activations).to(device)

# define loss function (criterion) and optimizer
# criterion = nn.BCEWithLogitsLoss().to(device)
criterion = nn.CrossEntropyLoss().to(device)

optimizer = torch.optim.Adam(model.parameters(),lr=lr,weight_decay=weight_decay)
```

Train model and validate

In [11]:

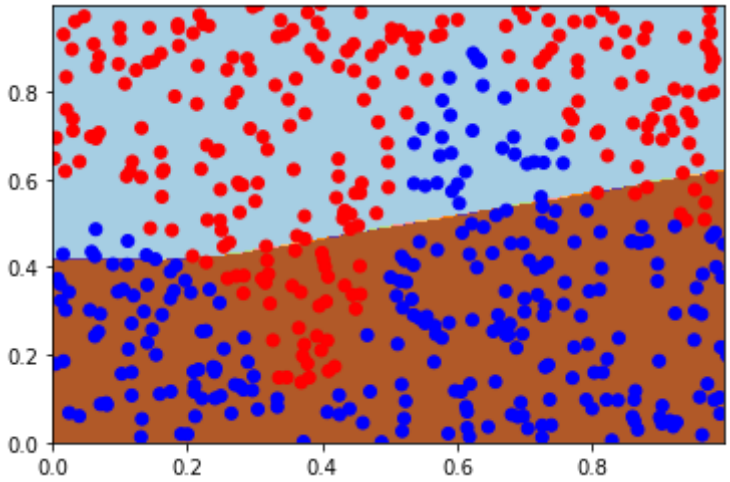
```
best_prec1 = 0
for epoch in range(num_epochs):
    if epoch in lr_step:
        for param_group in optimizer.param_groups:
            param_group['lr'] *= 0.1

    # train for one epoch
    train(train_loader, model, criterion, optimizer, epoch)

    # evaluate on validation set
    # prec1 = 0
    prec1 = validate(val_loader, model, criterion)

    # remember best prec@1 and save checkpoint
    is_best = prec1 > best_prec1
    best_prec1 = max(prec1, best_prec1)
    save_checkpoint({
        'epoch': epoch + 1,
        'state_dict': model.state_dict(),
        'best_prec1': best_prec1,
        'optimizer': optimizer.state_dict(),
    }, is_best,filename="checkpoint.pth.tar")
    print("-----------------------------------------------")

    if epoch%print_freq==0:
        plot_decision_boundary(model)

plot_decision_boundary(model)
```

```
Epoch: [0/200][0/4]     LR: 0.01        Loss 0.6895 (0.6895)    Train Acc 57.031
(57.031)
 * Train Acc 51.000
Test: [0/2]     Loss 0.6871 (0.6871)    Prec@1 49.219 (49.219)
 * Test Acc 50.000
-----------------------------------------------
```



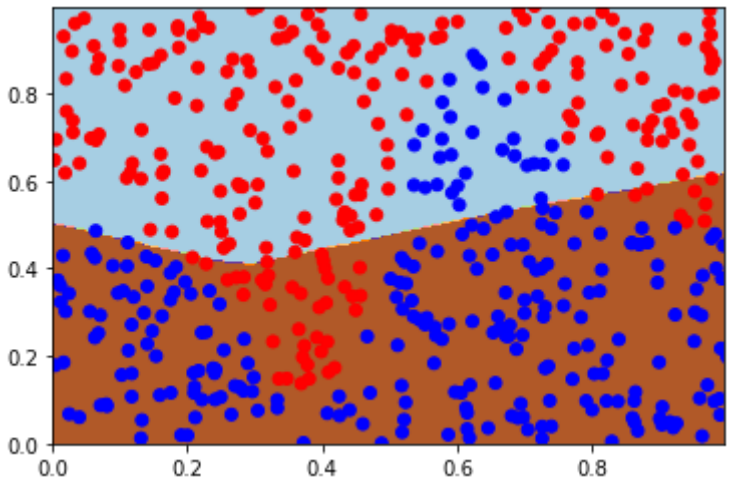

```
Epoch: [1/200][0/4]     LR: 0.01        Loss 0.6881 (0.6881)    Train Acc 48.828
(48.828)
 * Train Acc 65.000
Test: [0/2]     Loss 0.6783 (0.6783)    Prec@1 65.234 (65.234)
 * Test Acc 67.333
------------------------------------------------
Epoch: [2/200][0/4]     LR: 0.01        Loss 0.6792 (0.6792)    Train Acc 64.453
(64.453)
 * Train Acc 70.625
Test: [0/2]     Loss 0.6581 (0.6581)    Prec@1 81.250 (81.250)
 * Test Acc 81.000
------------------------------------------------
Epoch: [3/200][0/4]     LR: 0.01        Loss 0.6617 (0.6617)    Train Acc 78.906
(78.906)
 * Train Acc 75.625
Test: [0/2]     Loss 0.6222 (0.6222)    Prec@1 71.094 (71.094)
 * Test Acc 70.333
------------------------------------------------
Epoch: [4/200][0/4]     LR: 0.01        Loss 0.6265 (0.6265)    Train Acc 69.141
(69.141)
 * Train Acc 69.500
Test: [0/2]     Loss 0.5798 (0.5798)    Prec@1 73.828 (73.828)
 * Test Acc 74.667
------------------------------------------------
Epoch: [5/200][0/4]     LR: 0.01        Loss 0.5918 (0.5918)    Train Acc 73.438
(73.438)
 * Train Acc 75.625
Test: [0/2]     Loss 0.5192 (0.5192)    Prec@1 78.906 (78.906)
 * Test Acc 80.333
------------------------------------------------
Epoch: [6/200][0/4]     LR: 0.01        Loss 0.5361 (0.5361)    Train Acc 77.734
(77.734)
 * Train Acc 77.625
Test: [0/2]     Loss 0.4616 (0.4616)    Prec@1 82.812 (82.812)
 * Test Acc 82.000
------------------------------------------------
Epoch: [7/200][0/4]     LR: 0.01        Loss 0.5204 (0.5204)    Train Acc 75.391
(75.391)
 * Train Acc 78.875
Test: [0/2]     Loss 0.4327 (0.4327)    Prec@1 83.594 (83.594)
 * Test Acc 83.000
------------------------------------------------
Epoch: [8/200][0/4]     LR: 0.01        Loss 0.4706 (0.4706)    Train Acc 80.469
(80.469)
 * Train Acc 79.750
Test: [0/2]     Loss 0.4193 (0.4193)    Prec@1 83.203 (83.203)
 * Test Acc 84.000
------------------------------------------------
```

```
Epoch: [9/200][0/4]     LR: 0.01        Loss 0.4518 (0.4518)    Train Acc 79.688
(79.688)
 * Train Acc 80.000
Test: [0/2]     Loss 0.3855 (0.3855)    Prec@1 83.984 (83.984)
 * Test Acc 84.333
------------------------------------------------
Epoch: [10/200][0/4]    LR: 0.01        Loss 0.4496 (0.4496)    Train Acc 80.469
(80.469)
 * Train Acc 81.000
Test: [0/2]     Loss 0.3353 (0.3353)    Prec@1 85.938 (85.938)
 * Test Acc 85.000
------------------------------------------------
```



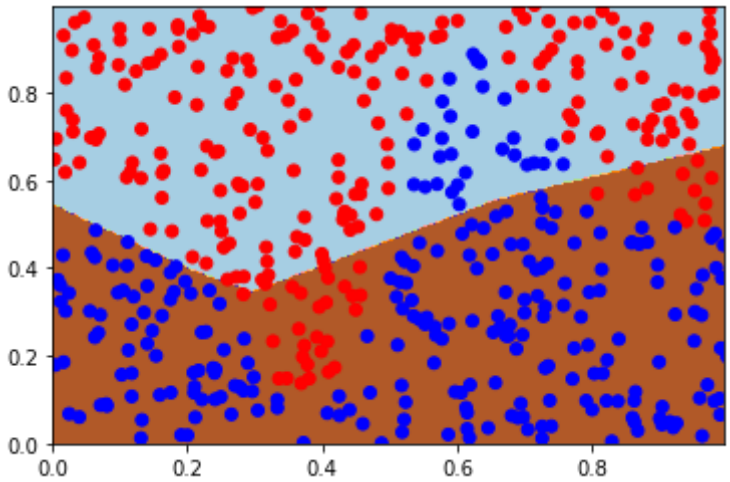

```
Epoch: [11/200][0/4]    LR: 0.01        Loss 0.3666 (0.3666)    Train Acc 85.156
(85.156)
 * Train Acc 82.000
Test: [0/2]     Loss 0.3594 (0.3594)    Prec@1 84.375 (84.375)
 * Test Acc 85.000
------------------------------------------------
Epoch: [12/200][0/4]    LR: 0.01        Loss 0.4055 (0.4055)    Train Acc 83.594
(83.594)
 * Train Acc 82.375
Test: [0/2]     Loss 0.3608 (0.3608)    Prec@1 85.547 (85.547)
 * Test Acc 86.000
------------------------------------------------
Epoch: [13/200][0/4]    LR: 0.01        Loss 0.4150 (0.4150)    Train Acc 81.641
(81.641)
 * Train Acc 82.375
Test: [0/2]     Loss 0.3427 (0.3427)    Prec@1 87.109 (87.109)
 * Test Acc 87.000
------------------------------------------------
Epoch: [14/200][0/4]    LR: 0.01        Loss 0.4777 (0.4777)    Train Acc 80.469
(80.469)
 * Train Acc 82.625
Test: [0/2]     Loss 0.3423 (0.3423)    Prec@1 87.891 (87.891)
 * Test Acc 87.667
------------------------------------------------
Epoch: [15/200][0/4]    LR: 0.01        Loss 0.3932 (0.3932)    Train Acc 85.547
(85.547)
 * Train Acc 83.250
Test: [0/2]     Loss 0.3327 (0.3327)    Prec@1 87.891 (87.891)
 * Test Acc 86.333
------------------------------------------------
Epoch: [16/200][0/4]    LR: 0.01        Loss 0.3920 (0.3920)    Train Acc 85.938
(85.938)
 * Train Acc 82.500
Test: [0/2]     Loss 0.3569 (0.3569)    Prec@1 85.156 (85.156)
 * Test Acc 85.333
------------------------------------------------
```

```
Epoch: [17/200][0/4]    LR: 0.01        Loss 0.4422 (0.4422)    Train Acc 79.297
(79.297)
 * Train Acc 81.750
Test: [0/2]     Loss 0.3619 (0.3619)    Prec@1 85.547 (85.547)
 * Test Acc 85.333
------------------------------------------------
Epoch: [18/200][0/4]    LR: 0.01        Loss 0.4507 (0.4507)    Train Acc 80.078
(80.078)
 * Train Acc 82.375
Test: [0/2]     Loss 0.3361 (0.3361)    Prec@1 87.109 (87.109)
 * Test Acc 86.667
------------------------------------------------
Epoch: [19/200][0/4]    LR: 0.01        Loss 0.4124 (0.4124)    Train Acc 83.594
(83.594)
 * Train Acc 82.875
Test: [0/2]     Loss 0.3437 (0.3437)    Prec@1 87.109 (87.109)
 * Test Acc 87.000
------------------------------------------------
Epoch: [20/200][0/4]    LR: 0.01        Loss 0.3863 (0.3863)    Train Acc 85.938
(85.938)
 * Train Acc 83.375
Test: [0/2]     Loss 0.3362 (0.3362)    Prec@1 87.109 (87.109)
 * Test Acc 87.000
------------------------------------------------
```



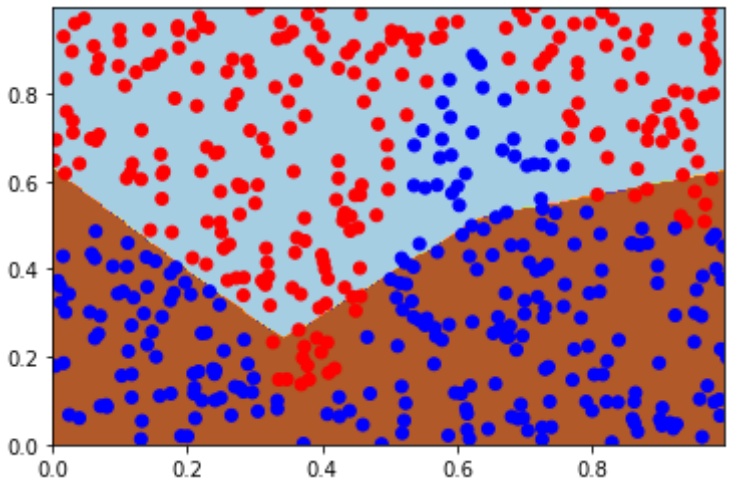

```
Epoch: [21/200][0/4]    LR: 0.01        Loss 0.3935 (0.3935)    Train Acc 84.375
(84.375)
 * Train Acc 83.375
Test: [0/2]     Loss 0.3559 (0.3559)    Prec@1 86.328 (86.328)
 * Test Acc 87.000
------------------------------------------------
Epoch: [22/200][0/4]    LR: 0.01        Loss 0.4117 (0.4117)    Train Acc 82.812
(82.812)
 * Train Acc 83.500
Test: [0/2]     Loss 0.3352 (0.3352)    Prec@1 87.500 (87.500)
 * Test Acc 87.000
------------------------------------------------
Epoch: [23/200][0/4]    LR: 0.01        Loss 0.3943 (0.3943)    Train Acc 82.812
(82.812)
 * Train Acc 83.500
Test: [0/2]     Loss 0.3619 (0.3619)    Prec@1 87.109 (87.109)
 * Test Acc 87.000
------------------------------------------------
Epoch: [24/200][0/4]    LR: 0.01        Loss 0.3675 (0.3675)    Train Acc 84.375
(84.375)
 * Train Acc 82.250
Test: [0/2]     Loss 0.3556 (0.3556)    Prec@1 83.984 (83.984)
 * Test Acc 85.000
------------------------------------------------
```

```
Epoch: [25/200][0/4]    LR: 0.01        Loss 0.3583 (0.3583)    Train Acc 86.719
(86.719)
 * Train Acc 82.125
Test: [0/2]     Loss 0.3720 (0.3720)    Prec@1 82.422 (82.422)
 * Test Acc 84.000
-----------------------------------------------
Epoch: [26/200][0/4]    LR: 0.01        Loss 0.4003 (0.4003)    Train Acc 82.812
(82.812)
 * Train Acc 83.375
Test: [0/2]     Loss 0.3639 (0.3639)    Prec@1 87.109 (87.109)
 * Test Acc 88.000
-----------------------------------------------
Epoch: [27/200][0/4]    LR: 0.01        Loss 0.3925 (0.3925)    Train Acc 84.766
(84.766)
 * Train Acc 84.500
Test: [0/2]     Loss 0.3426 (0.3426)    Prec@1 85.156 (85.156)
 * Test Acc 85.333
-----------------------------------------------
Epoch: [28/200][0/4]    LR: 0.01        Loss 0.3814 (0.3814)    Train Acc 83.984
(83.984)
 * Train Acc 84.125
Test: [0/2]     Loss 0.3209 (0.3209)    Prec@1 88.672 (88.672)
 * Test Acc 87.000
-----------------------------------------------
Epoch: [29/200][0/4]    LR: 0.01        Loss 0.4027 (0.4027)    Train Acc 82.031
(82.031)
 * Train Acc 85.000
Test: [0/2]     Loss 0.3352 (0.3352)    Prec@1 83.984 (83.984)
 * Test Acc 83.333
-----------------------------------------------
Epoch: [30/200][0/4]    LR: 0.01        Loss 0.3772 (0.3772)    Train Acc 84.766
(84.766)
 * Train Acc 83.125
Test: [0/2]     Loss 0.3252 (0.3252)    Prec@1 85.547 (85.547)
 * Test Acc 85.333
-----------------------------------------------
```

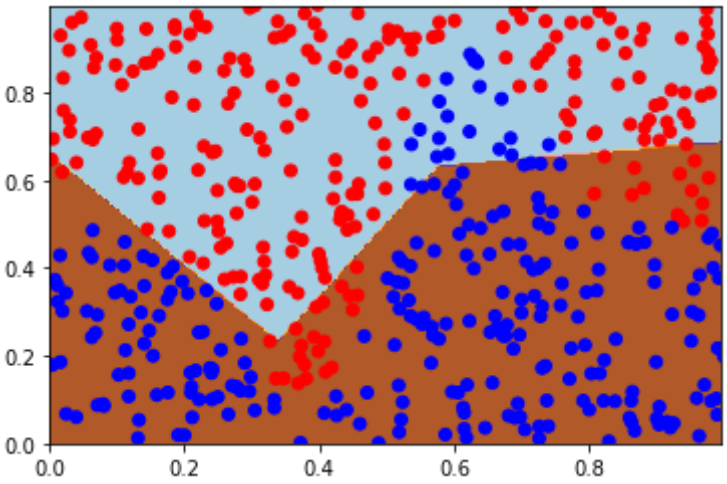



```
Epoch: [31/200][0/4]    LR: 0.01        Loss 0.4063 (0.4063)    Train Acc 83.594
(83.594)
 * Train Acc 84.875
Test: [0/2]     Loss 0.3167 (0.3167)    Prec@1 85.156 (85.156)
 * Test Acc 85.000
-----------------------------------------------
Epoch: [32/200][0/4]    LR: 0.01        Loss 0.3423 (0.3423)    Train Acc 85.938
(85.938)
 * Train Acc 84.750
Test: [0/2]     Loss 0.3516 (0.3516)    Prec@1 83.203 (83.203)
 * Test Acc 85.333
-----------------------------------------------
```

```
Epoch: [33/200][0/4]    LR: 0.01        Loss 0.3492 (0.3492)    Train Acc 87.109
(87.109)
 * Train Acc 84.375
Test: [0/2]     Loss 0.3334 (0.3334)    Prec@1 86.328 (86.328)
 * Test Acc 86.667
----------------------------------------------
Epoch: [34/200][0/4]    LR: 0.01        Loss 0.3765 (0.3765)    Train Acc 84.766
(84.766)
 * Train Acc 84.250
Test: [0/2]     Loss 0.3031 (0.3031)    Prec@1 85.547 (85.547)
 * Test Acc 84.000
----------------------------------------------
Epoch: [35/200][0/4]    LR: 0.01        Loss 0.3671 (0.3671)    Train Acc 84.766
(84.766)
 * Train Acc 83.125
Test: [0/2]     Loss 0.3198 (0.3198)    Prec@1 83.594 (83.594)
 * Test Acc 83.667
----------------------------------------------
Epoch: [36/200][0/4]    LR: 0.01        Loss 0.3894 (0.3894)    Train Acc 82.422
(82.422)
 * Train Acc 85.000
Test: [0/2]     Loss 0.3234 (0.3234)    Prec@1 85.938 (85.938)
 * Test Acc 87.000
----------------------------------------------
Epoch: [37/200][0/4]    LR: 0.01        Loss 0.3373 (0.3373)    Train Acc 86.328
(86.328)
 * Train Acc 85.500
Test: [0/2]     Loss 0.3327 (0.3327)    Prec@1 83.984 (83.984)
 * Test Acc 84.667
----------------------------------------------
Epoch: [38/200][0/4]    LR: 0.01        Loss 0.3434 (0.3434)    Train Acc 85.938
(85.938)
 * Train Acc 84.875
Test: [0/2]     Loss 0.3337 (0.3337)    Prec@1 85.156 (85.156)
 * Test Acc 86.667
----------------------------------------------
Epoch: [39/200][0/4]    LR: 0.01        Loss 0.4161 (0.4161)    Train Acc 81.641
(81.641)
 * Train Acc 85.125
Test: [0/2]     Loss 0.3237 (0.3237)    Prec@1 85.547 (85.547)
 * Test Acc 86.000
----------------------------------------------
Epoch: [40/200][0/4]    LR: 0.01        Loss 0.3493 (0.3493)    Train Acc 87.109
(87.109)
 * Train Acc 86.000
Test: [0/2]     Loss 0.3304 (0.3304)    Prec@1 85.156 (85.156)
 * Test Acc 86.000
----------------------------------------------
```

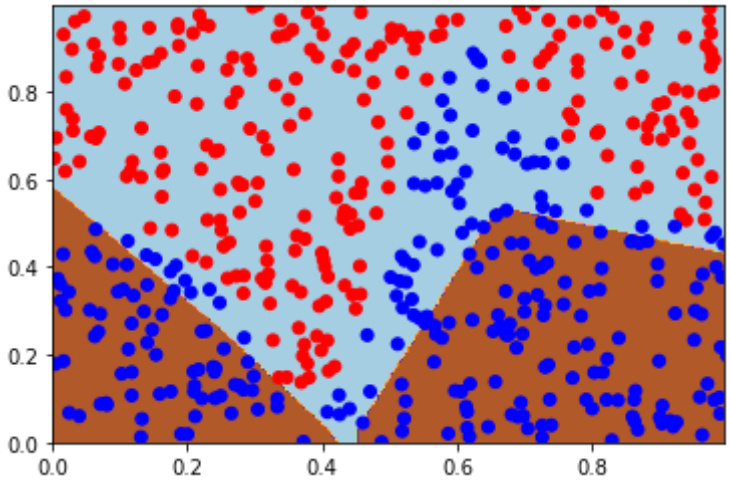


```
Epoch: [41/200][0/4]    LR: 0.01        Loss 0.3191 (0.3191)    Train Acc 85.938
(85.938)
 * Train Acc 83.750
Test: [0/2]     Loss 0.3255 (0.3255)    Prec@1 83.594 (83.594)
 * Test Acc 84.667
------------------------------------------------
Epoch: [42/200][0/4]    LR: 0.01        Loss 0.3021 (0.3021)    Train Acc 87.891
(87.891)
 * Train Acc 86.625
Test: [0/2]     Loss 0.3026 (0.3026)    Prec@1 90.234 (90.234)
 * Test Acc 88.667
------------------------------------------------
Epoch: [43/200][0/4]    LR: 0.01        Loss 0.3739 (0.3739)    Train Acc 85.156
(85.156)
 * Train Acc 85.625
Test: [0/2]     Loss 0.3147 (0.3147)    Prec@1 85.156 (85.156)
 * Test Acc 83.667
------------------------------------------------
Epoch: [44/200][0/4]    LR: 0.01        Loss 0.3460 (0.3460)    Train Acc 84.766
(84.766)
 * Train Acc 84.750
Test: [0/2]     Loss 0.3212 (0.3212)    Prec@1 85.938 (85.938)
 * Test Acc 87.000
------------------------------------------------
Epoch: [45/200][0/4]    LR: 0.01        Loss 0.3411 (0.3411)    Train Acc 85.938
(85.938)
 * Train Acc 86.875
Test: [0/2]     Loss 0.3029 (0.3029)    Prec@1 87.500 (87.500)
 * Test Acc 86.000
------------------------------------------------
Epoch: [46/200][0/4]    LR: 0.01        Loss 0.3171 (0.3171)    Train Acc 86.719
(86.719)
 * Train Acc 86.750
Test: [0/2]     Loss 0.2985 (0.2985)    Prec@1 86.328 (86.328)
 * Test Acc 86.000
------------------------------------------------
Epoch: [47/200][0/4]    LR: 0.01        Loss 0.3168 (0.3168)    Train Acc 87.500
(87.500)
 * Train Acc 86.625
Test: [0/2]     Loss 0.3144 (0.3144)    Prec@1 84.766 (84.766)
 * Test Acc 85.667
------------------------------------------------
Epoch: [48/200][0/4]    LR: 0.01        Loss 0.2954 (0.2954)    Train Acc 86.719
(86.719)
 * Train Acc 86.375
Test: [0/2]     Loss 0.3020 (0.3020)    Prec@1 86.328 (86.328)
 * Test Acc 86.000
------------------------------------------------
Epoch: [49/200][0/4]    LR: 0.01        Loss 0.3524 (0.3524)    Train Acc 83.984
(83.984)
 * Train Acc 86.250
Test: [0/2]     Loss 0.3020 (0.3020)    Prec@1 87.891 (87.891)
 * Test Acc 87.667
------------------------------------------------
Epoch: [50/200][0/4]    LR: 0.01        Loss 0.3363 (0.3363)    Train Acc 88.281
(88.281)
 * Train Acc 87.750
Test: [0/2]     Loss 0.3071 (0.3071)    Prec@1 83.594 (83.594)
 * Test Acc 84.667
------------------------------------------------
```



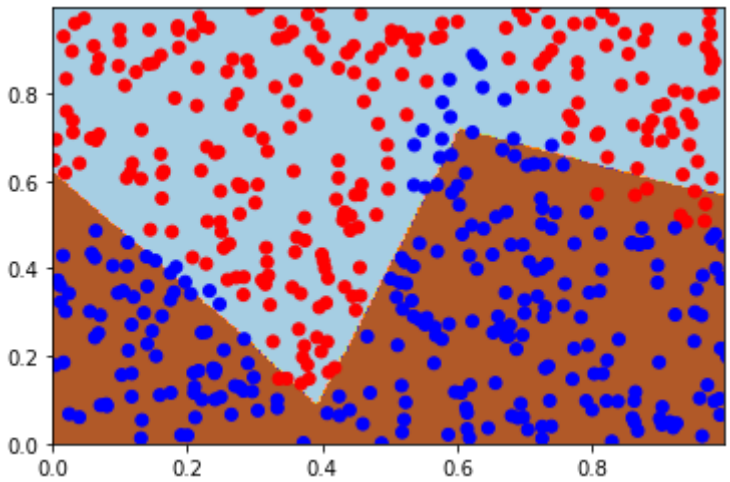

```
Epoch: [51/200][0/4]    LR: 0.01        Loss 0.2938 (0.2938)    Train Acc 89.453
(89.453)
 * Train Acc 86.375
Test: [0/2]     Loss 0.2873 (0.2873)    Prec@1 87.109 (87.109)
 * Test Acc 87.000
------------------------------------------------
Epoch: [52/200][0/4]    LR: 0.01        Loss 0.3515 (0.3515)    Train Acc 85.938
(85.938)
 * Train Acc 88.500
Test: [0/2]     Loss 0.2907 (0.2907)    Prec@1 85.547 (85.547)
 * Test Acc 85.333
------------------------------------------------
Epoch: [53/200][0/4]    LR: 0.01        Loss 0.2822 (0.2822)    Train Acc 87.891
(87.891)
 * Train Acc 86.500
Test: [0/2]     Loss 0.2838 (0.2838)    Prec@1 87.500 (87.500)
 * Test Acc 88.333
------------------------------------------------
Epoch: [54/200][0/4]    LR: 0.01        Loss 0.2873 (0.2873)    Train Acc 89.062
(89.062)
 * Train Acc 87.625
Test: [0/2]     Loss 0.2937 (0.2937)    Prec@1 84.766 (84.766)
 * Test Acc 85.000
------------------------------------------------
Epoch: [55/200][0/4]    LR: 0.01        Loss 0.3431 (0.3431)    Train Acc 85.156
(85.156)
 * Train Acc 85.250
Test: [0/2]     Loss 0.2479 (0.2479)    Prec@1 88.672 (88.672)
 * Test Acc 88.000
------------------------------------------------
Epoch: [56/200][0/4]    LR: 0.01        Loss 0.3031 (0.3031)    Train Acc 89.453
(89.453)
 * Train Acc 88.750
Test: [0/2]     Loss 0.2564 (0.2564)    Prec@1 89.453 (89.453)
 * Test Acc 90.000
------------------------------------------------
Epoch: [57/200][0/4]    LR: 0.01        Loss 0.2640 (0.2640)    Train Acc 91.016
(91.016)
 * Train Acc 89.625
Test: [0/2]     Loss 0.2671 (0.2671)    Prec@1 86.328 (86.328)
 * Test Acc 86.333
------------------------------------------------
Epoch: [58/200][0/4]    LR: 0.01        Loss 0.2855 (0.2855)    Train Acc 87.500
(87.500)
 * Train Acc 89.125
Test: [0/2]     Loss 0.2511 (0.2511)    Prec@1 89.844 (89.844)
 * Test Acc 89.667
------------------------------------------------
```

```
Epoch: [59/200][0/4]    LR: 0.01        Loss 0.2102 (0.2102)    Train Acc 93.750
(93.750)
 * Train Acc 89.375
Test: [0/2]     Loss 0.3052 (0.3052)    Prec@1 84.766 (84.766)
 * Test Acc 84.333
------------------------------------------------
Epoch: [60/200][0/4]    LR: 0.01        Loss 0.2868 (0.2868)    Train Acc 86.719
(86.719)
 * Train Acc 85.875
Test: [0/2]     Loss 0.2930 (0.2930)    Prec@1 87.891 (87.891)
 * Test Acc 88.333
------------------------------------------------
```



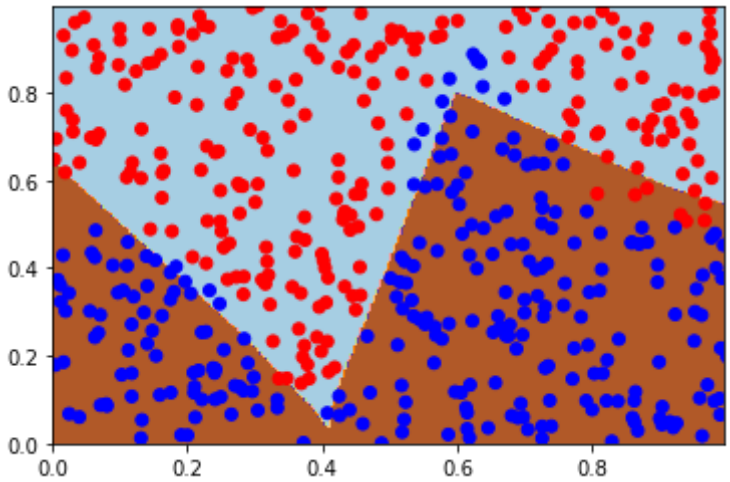

```
Epoch: [61/200][0/4]    LR: 0.01        Loss 0.2919 (0.2919)    Train Acc 88.281
(88.281)
 * Train Acc 86.500
Test: [0/2]     Loss 0.3293 (0.3293)    Prec@1 83.594 (83.594)
 * Test Acc 83.000
------------------------------------------------
Epoch: [62/200][0/4]    LR: 0.01        Loss 0.3266 (0.3266)    Train Acc 84.766
(84.766)
 * Train Acc 84.875
Test: [0/2]     Loss 0.2886 (0.2886)    Prec@1 88.672 (88.672)
 * Test Acc 89.000
------------------------------------------------
Epoch: [63/200][0/4]    LR: 0.01        Loss 0.2612 (0.2612)    Train Acc 88.672
(88.672)
 * Train Acc 86.625
Test: [0/2]     Loss 0.3383 (0.3383)    Prec@1 82.031 (82.031)
 * Test Acc 83.000
------------------------------------------------
Epoch: [64/200][0/4]    LR: 0.01        Loss 0.3040 (0.3040)    Train Acc 83.594
(83.594)
 * Train Acc 82.750
Test: [0/2]     Loss 0.2151 (0.2151)    Prec@1 94.141 (94.141)
 * Test Acc 92.667
------------------------------------------------
Epoch: [65/200][0/4]    LR: 0.01        Loss 0.2487 (0.2487)    Train Acc 90.234
(90.234)
 * Train Acc 86.625
Test: [0/2]     Loss 0.2291 (0.2291)    Prec@1 91.016 (91.016)
 * Test Acc 91.000
------------------------------------------------
Epoch: [66/200][0/4]    LR: 0.01        Loss 0.2472 (0.2472)    Train Acc 92.188
(92.188)
 * Train Acc 89.375
Test: [0/2]     Loss 0.2361 (0.2361)    Prec@1 89.453 (89.453)
 * Test Acc 88.667
------------------------------------------------
```

```
Epoch: [67/200][0/4]    LR: 0.01        Loss 0.2073 (0.2073)    Train Acc 93.359
(93.359)
 * Train Acc 89.125
Test: [0/2]     Loss 0.2409 (0.2409)    Prec@1 90.625 (90.625)
 * Test Acc 91.333
------------------------------------------------
Epoch: [68/200][0/4]    LR: 0.01        Loss 0.2617 (0.2617)    Train Acc 90.625
(90.625)
 * Train Acc 89.250
Test: [0/2]     Loss 0.2866 (0.2866)    Prec@1 86.328 (86.328)
 * Test Acc 86.000
------------------------------------------------
Epoch: [69/200][0/4]    LR: 0.01        Loss 0.3153 (0.3153)    Train Acc 84.766
(84.766)
 * Train Acc 87.500
Test: [0/2]     Loss 0.2210 (0.2210)    Prec@1 92.969 (92.969)
 * Test Acc 92.000
------------------------------------------------
Epoch: [70/200][0/4]    LR: 0.01        Loss 0.2784 (0.2784)    Train Acc 88.281
(88.281)
 * Train Acc 89.625
Test: [0/2]     Loss 0.2166 (0.2166)    Prec@1 90.625 (90.625)
 * Test Acc 89.667
------------------------------------------------
```



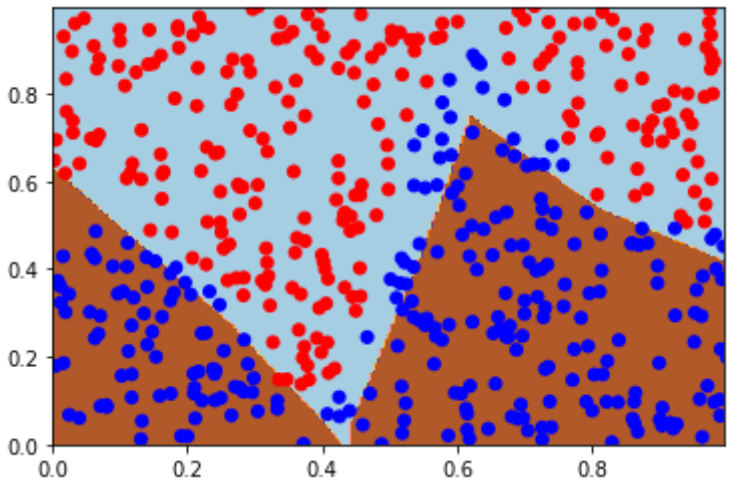

```
Epoch: [71/200][0/4]    LR: 0.01        Loss 0.2492 (0.2492)    Train Acc 92.188
(92.188)
 * Train Acc 90.750
Test: [0/2]     Loss 0.2289 (0.2289)    Prec@1 91.016 (91.016)
 * Test Acc 91.333
------------------------------------------------
Epoch: [72/200][0/4]    LR: 0.01        Loss 0.2870 (0.2870)    Train Acc 87.109
(87.109)
 * Train Acc 90.125
Test: [0/2]     Loss 0.2240 (0.2240)    Prec@1 91.797 (91.797)
 * Test Acc 91.667
------------------------------------------------
Epoch: [73/200][0/4]    LR: 0.01        Loss 0.2457 (0.2457)    Train Acc 91.406
(91.406)
 * Train Acc 90.250
Test: [0/2]     Loss 0.2154 (0.2154)    Prec@1 91.016 (91.016)
 * Test Acc 91.000
------------------------------------------------
Epoch: [74/200][0/4]    LR: 0.01        Loss 0.2556 (0.2556)    Train Acc 92.188
(92.188)
 * Train Acc 90.250
Test: [0/2]     Loss 0.2240 (0.2240)    Prec@1 90.625 (90.625)
 * Test Acc 90.667
------------------------------------------------
```

```
Epoch: [75/200][0/4]    LR: 0.01        Loss 0.2164 (0.2164)    Train Acc 91.797
(91.797)
 * Train Acc 89.000
Test: [0/2]     Loss 0.2050 (0.2050)    Prec@1 92.188 (92.188)
 * Test Acc 91.000
------------------------------------------------
Epoch: [76/200][0/4]    LR: 0.01        Loss 0.2254 (0.2254)    Train Acc 93.359
(93.359)
 * Train Acc 89.750
Test: [0/2]     Loss 0.2106 (0.2106)    Prec@1 92.188 (92.188)
 * Test Acc 92.000
------------------------------------------------
Epoch: [77/200][0/4]    LR: 0.01        Loss 0.2722 (0.2722)    Train Acc 89.844
(89.844)
 * Train Acc 91.625
Test: [0/2]     Loss 0.2355 (0.2355)    Prec@1 89.062 (89.062)
 * Test Acc 88.667
------------------------------------------------
Epoch: [78/200][0/4]    LR: 0.01        Loss 0.2799 (0.2799)    Train Acc 88.281
(88.281)
 * Train Acc 91.125
Test: [0/2]     Loss 0.2133 (0.2133)    Prec@1 92.969 (92.969)
 * Test Acc 93.000
------------------------------------------------
Epoch: [79/200][0/4]    LR: 0.01        Loss 0.2948 (0.2948)    Train Acc 88.672
(88.672)
 * Train Acc 91.125
Test: [0/2]     Loss 0.1977 (0.1977)    Prec@1 92.969 (92.969)
 * Test Acc 92.333
------------------------------------------------
Epoch: [80/200][0/4]    LR: 0.01        Loss 0.2331 (0.2331)    Train Acc 92.188
(92.188)
 * Train Acc 92.375
Test: [0/2]     Loss 0.2174 (0.2174)    Prec@1 90.234 (90.234)
 * Test Acc 90.000
------------------------------------------------
```

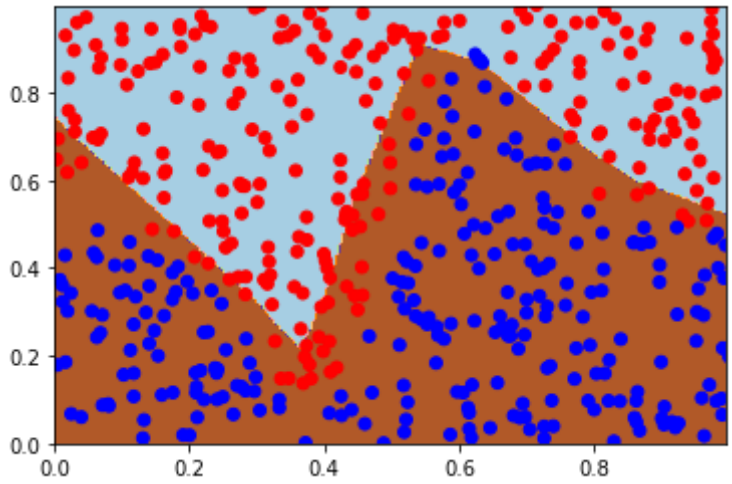



```
Epoch: [81/200][0/4]    LR: 0.01        Loss 0.2172 (0.2172)    Train Acc 90.234
(90.234)
 * Train Acc 91.625
Test: [0/2]     Loss 0.2306 (0.2306)    Prec@1 91.016 (91.016)
 * Test Acc 92.000
------------------------------------------------
Epoch: [82/200][0/4]    LR: 0.01        Loss 0.2229 (0.2229)    Train Acc 91.797
(91.797)
 * Train Acc 92.000
Test: [0/2]     Loss 0.2134 (0.2134)    Prec@1 92.578 (92.578)
 * Test Acc 93.000
------------------------------------------------
```

```
Epoch: [83/200][0/4]    LR: 0.01        Loss 0.2268 (0.2268)    Train Acc 91.797
(91.797)
 * Train Acc 91.250
Test: [0/2]     Loss 0.1930 (0.1930)    Prec@1 93.750 (93.750)
 * Test Acc 93.667
------------------------------------------------
Epoch: [84/200][0/4]    LR: 0.01        Loss 0.2506 (0.2506)    Train Acc 91.016
(91.016)
 * Train Acc 93.000
Test: [0/2]     Loss 0.1822 (0.1822)    Prec@1 94.531 (94.531)
 * Test Acc 94.000
------------------------------------------------
Epoch: [85/200][0/4]    LR: 0.01        Loss 0.2168 (0.2168)    Train Acc 94.531
(94.531)
 * Train Acc 93.750
Test: [0/2]     Loss 0.2086 (0.2086)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
------------------------------------------------
Epoch: [86/200][0/4]    LR: 0.01        Loss 0.2058 (0.2058)    Train Acc 93.359
(93.359)
 * Train Acc 93.375
Test: [0/2]     Loss 0.1818 (0.1818)    Prec@1 92.578 (92.578)
 * Test Acc 92.000
------------------------------------------------
Epoch: [87/200][0/4]    LR: 0.01        Loss 0.2239 (0.2239)    Train Acc 92.188
(92.188)
 * Train Acc 92.875
Test: [0/2]     Loss 0.1713 (0.1713)    Prec@1 92.578 (92.578)
 * Test Acc 92.333
------------------------------------------------
Epoch: [88/200][0/4]    LR: 0.01        Loss 0.2076 (0.2076)    Train Acc 92.578
(92.578)
 * Train Acc 92.750
Test: [0/2]     Loss 0.1849 (0.1849)    Prec@1 93.359 (93.359)
 * Test Acc 92.667
------------------------------------------------
Epoch: [89/200][0/4]    LR: 0.01        Loss 0.2024 (0.2024)    Train Acc 92.969
(92.969)
 * Train Acc 93.250
Test: [0/2]     Loss 0.1795 (0.1795)    Prec@1 95.703 (95.703)
 * Test Acc 95.000
------------------------------------------------
Epoch: [90/200][0/4]    LR: 0.01        Loss 0.2258 (0.2258)    Train Acc 92.969
(92.969)
 * Train Acc 93.000
Test: [0/2]     Loss 0.1861 (0.1861)    Prec@1 92.578 (92.578)
 * Test Acc 92.333
------------------------------------------------
```

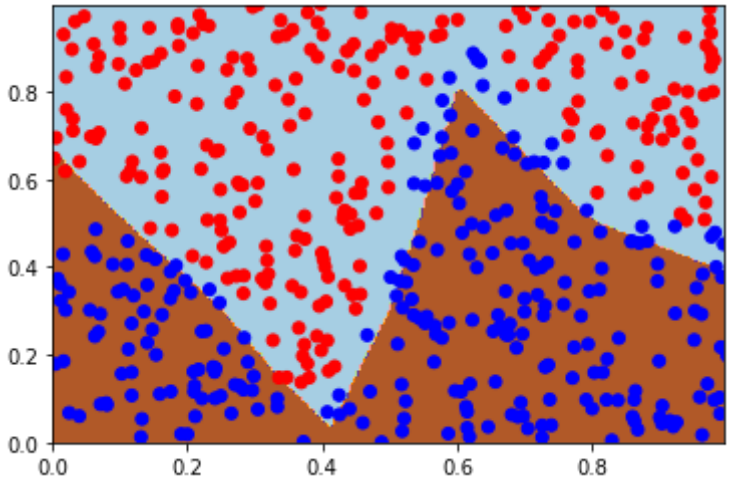


```
Epoch: [91/200][0/4]    LR: 0.01        Loss 0.2217 (0.2217)    Train Acc 90.234
(90.234)
 * Train Acc 91.500
Test: [0/2]     Loss 0.2251 (0.2251)    Prec@1 90.625 (90.625)
 * Test Acc 91.000
-----------------------------------------------
Epoch: [92/200][0/4]    LR: 0.01        Loss 0.2443 (0.2443)    Train Acc 92.578
(92.578)
 * Train Acc 92.250
Test: [0/2]     Loss 0.1795 (0.1795)    Prec@1 93.359 (93.359)
 * Test Acc 93.333
-----------------------------------------------
Epoch: [93/200][0/4]    LR: 0.01        Loss 0.2154 (0.2154)    Train Acc 92.188
(92.188)
 * Train Acc 93.250
Test: [0/2]     Loss 0.1688 (0.1688)    Prec@1 94.141 (94.141)
 * Test Acc 94.000
-----------------------------------------------
Epoch: [94/200][0/4]    LR: 0.01        Loss 0.2387 (0.2387)    Train Acc 92.969
(92.969)
 * Train Acc 92.500
Test: [0/2]     Loss 0.1984 (0.1984)    Prec@1 92.188 (92.188)
 * Test Acc 92.333
-----------------------------------------------
Epoch: [95/200][0/4]    LR: 0.01        Loss 0.2048 (0.2048)    Train Acc 93.750
(93.750)
 * Train Acc 91.750
Test: [0/2]     Loss 0.1608 (0.1608)    Prec@1 94.531 (94.531)
 * Test Acc 93.667
-----------------------------------------------
Epoch: [96/200][0/4]    LR: 0.01        Loss 0.2733 (0.2733)    Train Acc 88.281
(88.281)
 * Train Acc 91.250
Test: [0/2]     Loss 0.2310 (0.2310)    Prec@1 89.844 (89.844)
 * Test Acc 89.667
-----------------------------------------------
Epoch: [97/200][0/4]    LR: 0.01        Loss 0.2100 (0.2100)    Train Acc 92.188
(92.188)
 * Train Acc 92.375
Test: [0/2]     Loss 0.1881 (0.1881)    Prec@1 92.188 (92.188)
 * Test Acc 92.333
-----------------------------------------------
Epoch: [98/200][0/4]    LR: 0.01        Loss 0.2465 (0.2465)    Train Acc 87.500
(87.500)
 * Train Acc 90.375
Test: [0/2]     Loss 0.2663 (0.2663)    Prec@1 88.281 (88.281)
 * Test Acc 88.000
-----------------------------------------------
Epoch: [99/200][0/4]    LR: 0.01        Loss 0.2707 (0.2707)    Train Acc 89.453
(89.453)
 * Train Acc 91.500
Test: [0/2]     Loss 0.1837 (0.1837)    Prec@1 92.188 (92.188)
 * Test Acc 92.333
-----------------------------------------------
Epoch: [100/200][0/4]   LR: 0.01        Loss 0.2072 (0.2072)    Train Acc 92.188
(92.188)
 * Train Acc 92.375
Test: [0/2]     Loss 0.2579 (0.2579)    Prec@1 89.844 (89.844)
 * Test Acc 89.000
-----------------------------------------------
```



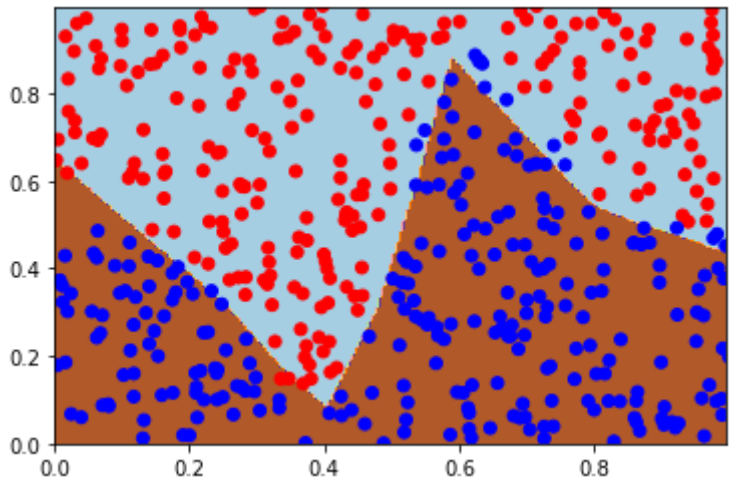

```
Epoch: [101/200][0/4]   LR: 0.01        Loss 0.2988 (0.2988)    Train Acc 84.766
(84.766)
 * Train Acc 89.375
Test: [0/2]     Loss 0.1627 (0.1627)    Prec@1 93.359 (93.359)
 * Test Acc 93.333
----------------------------------------------
Epoch: [102/200][0/4]   LR: 0.01        Loss 0.2088 (0.2088)    Train Acc 92.969
(92.969)
 * Train Acc 93.125
Test: [0/2]     Loss 0.2302 (0.2302)    Prec@1 87.500 (87.500)
 * Test Acc 88.000
----------------------------------------------
Epoch: [103/200][0/4]   LR: 0.01        Loss 0.2866 (0.2866)    Train Acc 89.844
(89.844)
 * Train Acc 91.750
Test: [0/2]     Loss 0.1699 (0.1699)    Prec@1 94.531 (94.531)
 * Test Acc 95.333
----------------------------------------------
Epoch: [104/200][0/4]   LR: 0.01        Loss 0.2062 (0.2062)    Train Acc 93.750
(93.750)
 * Train Acc 93.875
Test: [0/2]     Loss 0.1801 (0.1801)    Prec@1 94.922 (94.922)
 * Test Acc 95.667
----------------------------------------------
Epoch: [105/200][0/4]   LR: 0.01        Loss 0.2158 (0.2158)    Train Acc 95.312
(95.312)
 * Train Acc 93.000
Test: [0/2]     Loss 0.1816 (0.1816)    Prec@1 94.531 (94.531)
 * Test Acc 95.333
----------------------------------------------
Epoch: [106/200][0/4]   LR: 0.01        Loss 0.1772 (0.1772)    Train Acc 94.922
(94.922)
 * Train Acc 93.250
Test: [0/2]     Loss 0.1628 (0.1628)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
----------------------------------------------
Epoch: [107/200][0/4]   LR: 0.01        Loss 0.1997 (0.1997)    Train Acc 92.188
(92.188)
 * Train Acc 91.000
Test: [0/2]     Loss 0.2229 (0.2229)    Prec@1 91.406 (91.406)
 * Test Acc 91.667
----------------------------------------------
Epoch: [108/200][0/4]   LR: 0.01        Loss 0.1716 (0.1716)    Train Acc 92.188
(92.188)
 * Train Acc 91.125
Test: [0/2]     Loss 0.1528 (0.1528)    Prec@1 94.531 (94.531)
 * Test Acc 94.333
----------------------------------------------
```

```
Epoch: [109/200][0/4]   LR: 0.01        Loss 0.2061 (0.2061)    Train Acc 92.969
(92.969)
 * Train Acc 94.250
Test: [0/2]     Loss 0.1976 (0.1976)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
------------------------------------------------
Epoch: [110/200][0/4]   LR: 0.01        Loss 0.2079 (0.2079)    Train Acc 93.359
(93.359)
 * Train Acc 94.375
Test: [0/2]     Loss 0.1794 (0.1794)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
------------------------------------------------
```



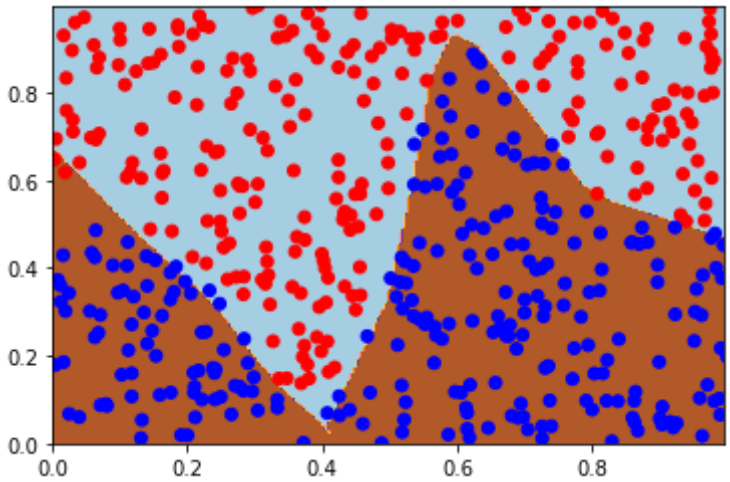

```
Epoch: [111/200][0/4]   LR: 0.01        Loss 0.2750 (0.2750)    Train Acc 89.453
(89.453)
 * Train Acc 90.750
Test: [0/2]     Loss 0.2295 (0.2295)    Prec@1 91.016 (91.016)
 * Test Acc 91.333
------------------------------------------------
Epoch: [112/200][0/4]   LR: 0.01        Loss 0.2518 (0.2518)    Train Acc 91.016
(91.016)
 * Train Acc 91.375
Test: [0/2]     Loss 0.2077 (0.2077)    Prec@1 91.406 (91.406)
 * Test Acc 92.667
------------------------------------------------
Epoch: [113/200][0/4]   LR: 0.01        Loss 0.3557 (0.3557)    Train Acc 86.328
(86.328)
 * Train Acc 90.375
Test: [0/2]     Loss 0.1882 (0.1882)    Prec@1 91.406 (91.406)
 * Test Acc 91.667
------------------------------------------------
Epoch: [114/200][0/4]   LR: 0.01        Loss 0.2188 (0.2188)    Train Acc 94.141
(94.141)
 * Train Acc 92.500
Test: [0/2]     Loss 0.1672 (0.1672)    Prec@1 94.531 (94.531)
 * Test Acc 95.000
------------------------------------------------
Epoch: [115/200][0/4]   LR: 0.01        Loss 0.2274 (0.2274)    Train Acc 91.797
(91.797)
 * Train Acc 94.000
Test: [0/2]     Loss 0.1708 (0.1708)    Prec@1 94.531 (94.531)
 * Test Acc 94.667
------------------------------------------------
Epoch: [116/200][0/4]   LR: 0.01        Loss 0.2289 (0.2289)    Train Acc 92.188
(92.188)
 * Train Acc 93.375
Test: [0/2]     Loss 0.1538 (0.1538)    Prec@1 96.094 (96.094)
 * Test Acc 95.667
------------------------------------------------
```

```
Epoch: [117/200][0/4]   LR: 0.01        Loss 0.1992 (0.1992)    Train Acc 95.312
(95.312)
 * Train Acc 95.250
Test: [0/2]     Loss 0.1634 (0.1634)    Prec@1 93.750 (93.750)
 * Test Acc 94.000
------------------------------------------------
Epoch: [118/200][0/4]   LR: 0.01        Loss 0.2079 (0.2079)    Train Acc 94.922
(94.922)
 * Train Acc 94.375
Test: [0/2]     Loss 0.1587 (0.1587)    Prec@1 93.750 (93.750)
 * Test Acc 94.333
------------------------------------------------
Epoch: [119/200][0/4]   LR: 0.01        Loss 0.1386 (0.1386)    Train Acc 96.875
(96.875)
 * Train Acc 95.125
Test: [0/2]     Loss 0.1584 (0.1584)    Prec@1 94.922 (94.922)
 * Test Acc 95.000
------------------------------------------------
Epoch: [120/200][0/4]   LR: 0.01        Loss 0.1787 (0.1787)    Train Acc 93.750
(93.750)
 * Train Acc 93.625
Test: [0/2]     Loss 0.1458 (0.1458)    Prec@1 95.312 (95.312)
 * Test Acc 95.000
------------------------------------------------
```



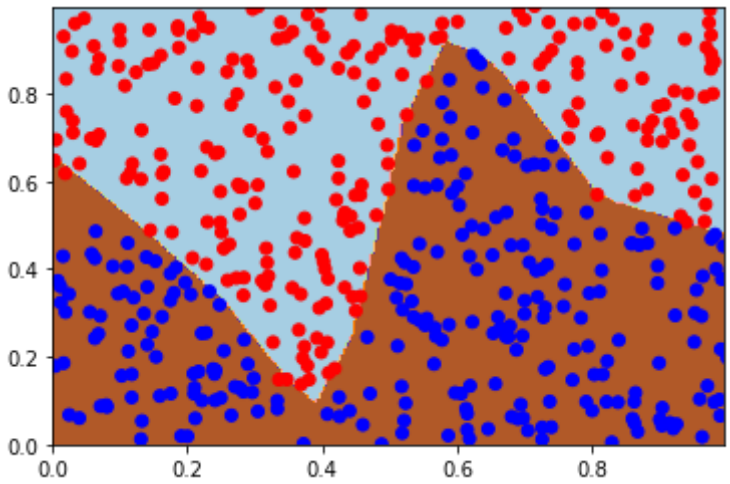

```
Epoch: [121/200][0/4]   LR: 0.01        Loss 0.2068 (0.2068)    Train Acc 93.359
(93.359)
 * Train Acc 94.250
Test: [0/2]     Loss 0.1599 (0.1599)    Prec@1 94.922 (94.922)
 * Test Acc 94.667
------------------------------------------------
Epoch: [122/200][0/4]   LR: 0.01        Loss 0.2036 (0.2036)    Train Acc 94.922
(94.922)
 * Train Acc 94.000
Test: [0/2]     Loss 0.1412 (0.1412)    Prec@1 95.703 (95.703)
 * Test Acc 95.000
------------------------------------------------
Epoch: [123/200][0/4]   LR: 0.01        Loss 0.2096 (0.2096)    Train Acc 94.922
(94.922)
 * Train Acc 95.250
Test: [0/2]     Loss 0.1658 (0.1658)    Prec@1 96.094 (96.094)
 * Test Acc 96.667
------------------------------------------------
Epoch: [124/200][0/4]   LR: 0.01        Loss 0.1898 (0.1898)    Train Acc 93.359
(93.359)
 * Train Acc 93.625
Test: [0/2]     Loss 0.1586 (0.1586)    Prec@1 94.922 (94.922)
 * Test Acc 95.333
------------------------------------------------
```

```
Epoch: [125/200][0/4]   LR: 0.01        Loss 0.1836 (0.1836)    Train Acc 95.703
(95.703)
 * Train Acc 94.625
Test: [0/2]     Loss 0.1436 (0.1436)    Prec@1 95.703 (95.703)
 * Test Acc 95.667
-------------------------------------------------
Epoch: [126/200][0/4]   LR: 0.01        Loss 0.2246 (0.2246)    Train Acc 94.922
(94.922)
 * Train Acc 95.125
Test: [0/2]     Loss 0.1383 (0.1383)    Prec@1 96.484 (96.484)
 * Test Acc 96.000
-------------------------------------------------
Epoch: [127/200][0/4]   LR: 0.01        Loss 0.1640 (0.1640)    Train Acc 96.484
(96.484)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1621 (0.1621)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
-------------------------------------------------
Epoch: [128/200][0/4]   LR: 0.01        Loss 0.2121 (0.2121)    Train Acc 90.625
(90.625)
 * Train Acc 92.000
Test: [0/2]     Loss 0.1991 (0.1991)    Prec@1 92.578 (92.578)
 * Test Acc 92.667
-------------------------------------------------
Epoch: [129/200][0/4]   LR: 0.01        Loss 0.1820 (0.1820)    Train Acc 93.359
(93.359)
 * Train Acc 94.375
Test: [0/2]     Loss 0.1715 (0.1715)    Prec@1 93.359 (93.359)
 * Test Acc 94.333
-------------------------------------------------
Epoch: [130/200][0/4]   LR: 0.01        Loss 0.1998 (0.1998)    Train Acc 91.016
(91.016)
 * Train Acc 92.625
Test: [0/2]     Loss 0.1565 (0.1565)    Prec@1 96.094 (96.094)
 * Test Acc 95.667
-------------------------------------------------
```

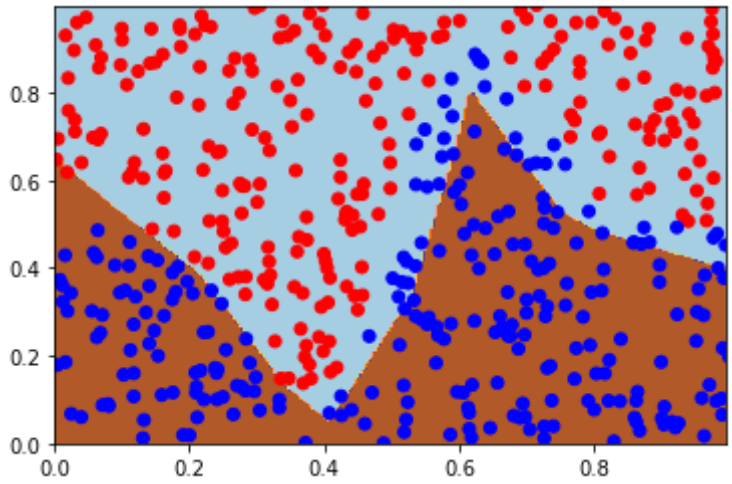



```
Epoch: [131/200][0/4]   LR: 0.01        Loss 0.1785 (0.1785)    Train Acc 95.703
(95.703)
 * Train Acc 92.875
Test: [0/2]     Loss 0.2132 (0.2132)    Prec@1 90.625 (90.625)
 * Test Acc 91.667
-------------------------------------------------
Epoch: [132/200][0/4]   LR: 0.01        Loss 0.1861 (0.1861)    Train Acc 94.141
(94.141)
 * Train Acc 92.375
Test: [0/2]     Loss 0.1684 (0.1684)    Prec@1 92.578 (92.578)
 * Test Acc 92.333
-------------------------------------------------
```

```
Epoch: [133/200][0/4]   LR: 0.01        Loss 0.1775 (0.1775)    Train Acc 92.188
(92.188)
 * Train Acc 91.250
Test: [0/2]     Loss 0.2097 (0.2097)    Prec@1 90.234 (90.234)
 * Test Acc 90.667
------------------------------------------------
Epoch: [134/200][0/4]   LR: 0.01        Loss 0.2027 (0.2027)    Train Acc 92.188
(92.188)
 * Train Acc 93.000
Test: [0/2]     Loss 0.1404 (0.1404)    Prec@1 96.875 (96.875)
 * Test Acc 97.000
------------------------------------------------
Epoch: [135/200][0/4]   LR: 0.01        Loss 0.2530 (0.2530)    Train Acc 91.797
(91.797)
 * Train Acc 94.000
Test: [0/2]     Loss 0.1187 (0.1187)    Prec@1 96.875 (96.875)
 * Test Acc 96.667
------------------------------------------------
Epoch: [136/200][0/4]   LR: 0.01        Loss 0.2445 (0.2445)    Train Acc 94.141
(94.141)
 * Train Acc 91.625
Test: [0/2]     Loss 0.1516 (0.1516)    Prec@1 96.094 (96.094)
 * Test Acc 95.667
------------------------------------------------
Epoch: [137/200][0/4]   LR: 0.01        Loss 0.1750 (0.1750)    Train Acc 94.922
(94.922)
 * Train Acc 92.375
Test: [0/2]     Loss 0.1497 (0.1497)    Prec@1 93.359 (93.359)
 * Test Acc 93.333
------------------------------------------------
Epoch: [138/200][0/4]   LR: 0.01        Loss 0.1980 (0.1980)    Train Acc 92.969
(92.969)
 * Train Acc 92.250
Test: [0/2]     Loss 0.1599 (0.1599)    Prec@1 94.922 (94.922)
 * Test Acc 95.000
------------------------------------------------
Epoch: [139/200][0/4]   LR: 0.01        Loss 0.2224 (0.2224)    Train Acc 94.531
(94.531)
 * Train Acc 94.250
Test: [0/2]     Loss 0.1437 (0.1437)    Prec@1 94.922 (94.922)
 * Test Acc 95.333
------------------------------------------------
Epoch: [140/200][0/4]   LR: 0.01        Loss 0.2223 (0.2223)    Train Acc 93.359
(93.359)
 * Train Acc 95.125
Test: [0/2]     Loss 0.1644 (0.1644)    Prec@1 96.484 (96.484)
 * Test Acc 96.667
------------------------------------------------
```

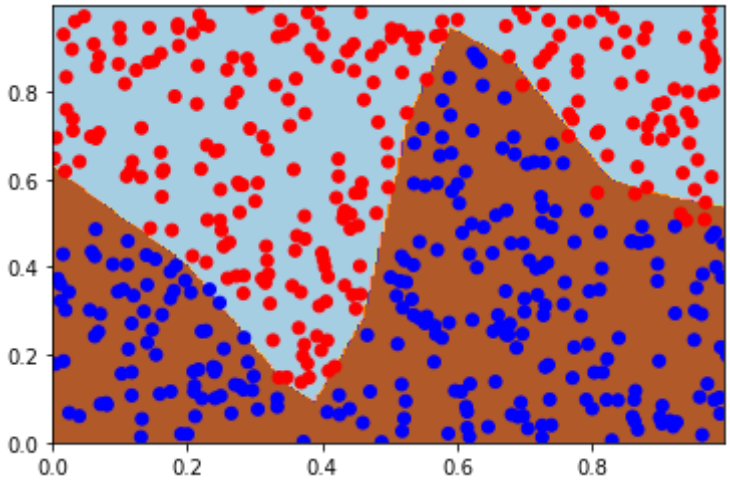


```
Epoch: [141/200][0/4]   LR: 0.01        Loss 0.2223 (0.2223)    Train Acc 95.312
(95.312)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1405 (0.1405)    Prec@1 95.312 (95.312)
 * Test Acc 95.333
------------------------------------------------
Epoch: [142/200][0/4]   LR: 0.01        Loss 0.2422 (0.2422)    Train Acc 90.625
(90.625)
 * Train Acc 94.000
Test: [0/2]     Loss 0.1619 (0.1619)    Prec@1 94.531 (94.531)
 * Test Acc 95.333
------------------------------------------------
Epoch: [143/200][0/4]   LR: 0.01        Loss 0.1492 (0.1492)    Train Acc 95.312
(95.312)
 * Train Acc 94.625
Test: [0/2]     Loss 0.1502 (0.1502)    Prec@1 95.312 (95.312)
 * Test Acc 95.333
------------------------------------------------
Epoch: [144/200][0/4]   LR: 0.01        Loss 0.2026 (0.2026)    Train Acc 94.922
(94.922)
 * Train Acc 94.875
Test: [0/2]     Loss 0.1336 (0.1336)    Prec@1 96.484 (96.484)
 * Test Acc 96.667
------------------------------------------------
Epoch: [145/200][0/4]   LR: 0.01        Loss 0.1880 (0.1880)    Train Acc 95.703
(95.703)
 * Train Acc 95.750
Test: [0/2]     Loss 0.1281 (0.1281)    Prec@1 94.531 (94.531)
 * Test Acc 94.333
------------------------------------------------
Epoch: [146/200][0/4]   LR: 0.01        Loss 0.2283 (0.2283)    Train Acc 92.969
(92.969)
 * Train Acc 92.875
Test: [0/2]     Loss 0.1588 (0.1588)    Prec@1 94.141 (94.141)
 * Test Acc 94.000
------------------------------------------------
Epoch: [147/200][0/4]   LR: 0.01        Loss 0.2267 (0.2267)    Train Acc 92.969
(92.969)
 * Train Acc 94.750
Test: [0/2]     Loss 0.1447 (0.1447)    Prec@1 95.703 (95.703)
 * Test Acc 95.667
------------------------------------------------
Epoch: [148/200][0/4]   LR: 0.01        Loss 0.2002 (0.2002)    Train Acc 94.922
(94.922)
 * Train Acc 94.875
Test: [0/2]     Loss 0.1355 (0.1355)    Prec@1 97.266 (97.266)
 * Test Acc 96.000
------------------------------------------------
Epoch: [149/200][0/4]   LR: 0.01        Loss 0.1799 (0.1799)    Train Acc 97.266
(97.266)
 * Train Acc 95.500
Test: [0/2]     Loss 0.1866 (0.1866)    Prec@1 94.922 (94.922)
 * Test Acc 95.333
------------------------------------------------
Epoch: [150/200][0/4]   LR: 0.01        Loss 0.1576 (0.1576)    Train Acc 95.312
(95.312)
 * Train Acc 92.750
Test: [0/2]     Loss 0.1720 (0.1720)    Prec@1 91.406 (91.406)
 * Test Acc 92.333
------------------------------------------------
```



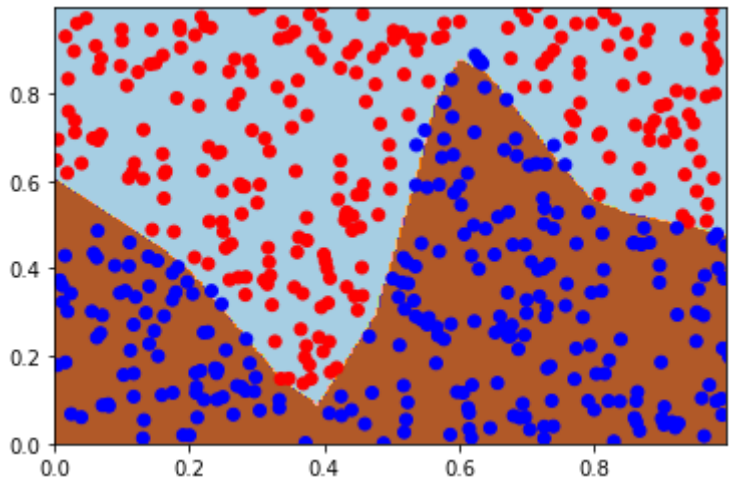

```
Epoch: [151/200][0/4]   LR: 0.01        Loss 0.1986 (0.1986)    Train Acc 91.797
(91.797)
 * Train Acc 90.875
Test: [0/2]     Loss 0.2326 (0.2326)    Prec@1 88.672 (88.672)
 * Test Acc 88.000
----------------------------------------------
Epoch: [152/200][0/4]   LR: 0.01        Loss 0.2126 (0.2126)    Train Acc 92.969
(92.969)
 * Train Acc 92.250
Test: [0/2]     Loss 0.1944 (0.1944)    Prec@1 91.797 (91.797)
 * Test Acc 91.667
----------------------------------------------
Epoch: [153/200][0/4]   LR: 0.01        Loss 0.1992 (0.1992)    Train Acc 89.844
(89.844)
 * Train Acc 89.500
Test: [0/2]     Loss 0.2632 (0.2632)    Prec@1 85.547 (85.547)
 * Test Acc 86.667
----------------------------------------------
Epoch: [154/200][0/4]   LR: 0.01        Loss 0.2595 (0.2595)    Train Acc 90.234
(90.234)
 * Train Acc 91.500
Test: [0/2]     Loss 0.2053 (0.2053)    Prec@1 92.188 (92.188)
 * Test Acc 92.667
----------------------------------------------
Epoch: [155/200][0/4]   LR: 0.01        Loss 0.2953 (0.2953)    Train Acc 87.500
(87.500)
 * Train Acc 90.750
Test: [0/2]     Loss 0.1766 (0.1766)    Prec@1 93.359 (93.359)
 * Test Acc 93.667
----------------------------------------------
Epoch: [156/200][0/4]   LR: 0.01        Loss 0.1716 (0.1716)    Train Acc 95.312
(95.312)
 * Train Acc 94.000
Test: [0/2]     Loss 0.1303 (0.1303)    Prec@1 93.750 (93.750)
 * Test Acc 93.667
----------------------------------------------
Epoch: [157/200][0/4]   LR: 0.01        Loss 0.2087 (0.2087)    Train Acc 92.188
(92.188)
 * Train Acc 93.875
Test: [0/2]     Loss 0.1434 (0.1434)    Prec@1 97.266 (97.266)
 * Test Acc 97.333
----------------------------------------------
Epoch: [158/200][0/4]   LR: 0.01        Loss 0.2376 (0.2376)    Train Acc 93.750
(93.750)
 * Train Acc 93.625
Test: [0/2]     Loss 0.1356 (0.1356)    Prec@1 96.484 (96.484)
 * Test Acc 96.000
----------------------------------------------
```

```
Epoch: [159/200][0/4]   LR: 0.01        Loss 0.1874 (0.1874)    Train Acc 95.312
(95.312)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1339 (0.1339)    Prec@1 94.141 (94.141)
 * Test Acc 94.000
------------------------------------------------
Epoch: [160/200][0/4]   LR: 0.01        Loss 0.2203 (0.2203)    Train Acc 89.844
(89.844)
 * Train Acc 91.625
Test: [0/2]     Loss 0.1572 (0.1572)    Prec@1 94.531 (94.531)
 * Test Acc 94.000
------------------------------------------------
```



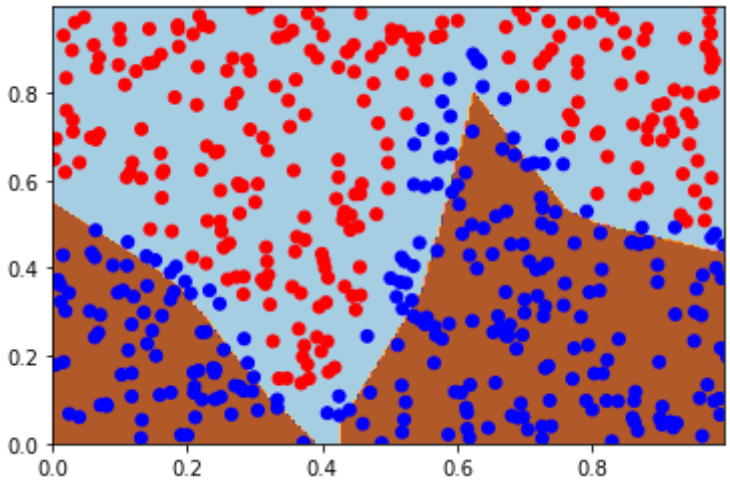

```
Epoch: [161/200][0/4]   LR: 0.01        Loss 0.1225 (0.1225)    Train Acc 97.656
(97.656)
 * Train Acc 93.875
Test: [0/2]     Loss 0.1374 (0.1374)    Prec@1 94.141 (94.141)
 * Test Acc 94.000
------------------------------------------------
Epoch: [162/200][0/4]   LR: 0.01        Loss 0.1622 (0.1622)    Train Acc 94.531
(94.531)
 * Train Acc 93.375
Test: [0/2]     Loss 0.1734 (0.1734)    Prec@1 96.094 (96.094)
 * Test Acc 96.000
------------------------------------------------
Epoch: [163/200][0/4]   LR: 0.01        Loss 0.2060 (0.2060)    Train Acc 92.578
(92.578)
 * Train Acc 93.000
Test: [0/2]     Loss 0.1490 (0.1490)    Prec@1 92.578 (92.578)
 * Test Acc 93.333
------------------------------------------------
Epoch: [164/200][0/4]   LR: 0.01        Loss 0.1429 (0.1429)    Train Acc 93.359
(93.359)
 * Train Acc 93.250
Test: [0/2]     Loss 0.1589 (0.1589)    Prec@1 93.750 (93.750)
 * Test Acc 94.000
------------------------------------------------
Epoch: [165/200][0/4]   LR: 0.01        Loss 0.1846 (0.1846)    Train Acc 95.703
(95.703)
 * Train Acc 95.250
Test: [0/2]     Loss 0.1329 (0.1329)    Prec@1 96.094 (96.094)
 * Test Acc 95.667
------------------------------------------------
Epoch: [166/200][0/4]   LR: 0.01        Loss 0.1913 (0.1913)    Train Acc 92.969
(92.969)
 * Train Acc 94.625
Test: [0/2]     Loss 0.1350 (0.1350)    Prec@1 98.047 (98.047)
 * Test Acc 97.000
------------------------------------------------
```

```
Epoch: [167/200][0/4]   LR: 0.01        Loss 0.1666 (0.1666)    Train Acc 96.875
(96.875)
 * Train Acc 95.250
Test: [0/2]     Loss 0.1329 (0.1329)    Prec@1 94.531 (94.531)
 * Test Acc 94.000
------------------------------------------------
Epoch: [168/200][0/4]   LR: 0.01        Loss 0.1240 (0.1240)    Train Acc 95.312
(95.312)
 * Train Acc 93.625
Test: [0/2]     Loss 0.1568 (0.1568)    Prec@1 94.922 (94.922)
 * Test Acc 95.667
------------------------------------------------
Epoch: [169/200][0/4]   LR: 0.01        Loss 0.1926 (0.1926)    Train Acc 94.531
(94.531)
 * Train Acc 94.750
Test: [0/2]     Loss 0.1253 (0.1253)    Prec@1 94.531 (94.531)
 * Test Acc 95.000
------------------------------------------------
Epoch: [170/200][0/4]   LR: 0.01        Loss 0.1540 (0.1540)    Train Acc 94.531
(94.531)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1089 (0.1089)    Prec@1 96.484 (96.484)
 * Test Acc 96.000
------------------------------------------------
```



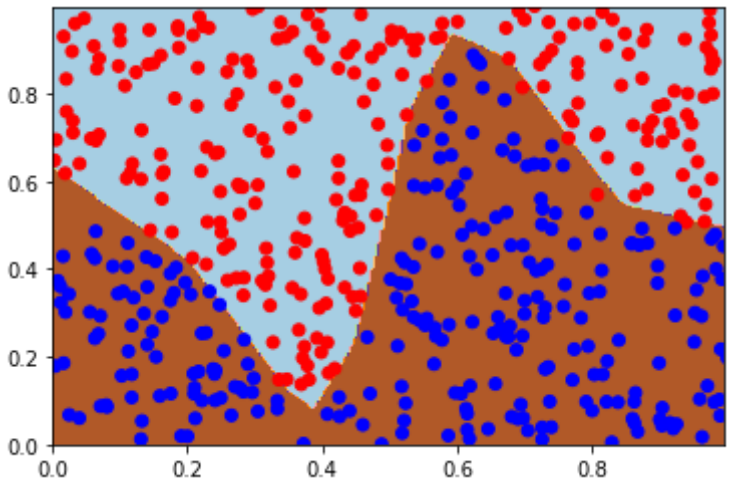

```
Epoch: [171/200][0/4]   LR: 0.01        Loss 0.1604 (0.1604)    Train Acc 93.750
(93.750)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1302 (0.1302)    Prec@1 95.703 (95.703)
 * Test Acc 96.000
------------------------------------------------
Epoch: [172/200][0/4]   LR: 0.01        Loss 0.2062 (0.2062)    Train Acc 95.703
(95.703)
 * Train Acc 96.125
Test: [0/2]     Loss 0.1355 (0.1355)    Prec@1 93.750 (93.750)
 * Test Acc 94.000
------------------------------------------------
Epoch: [173/200][0/4]   LR: 0.01        Loss 0.1752 (0.1752)    Train Acc 92.969
(92.969)
 * Train Acc 93.750
Test: [0/2]     Loss 0.1483 (0.1483)    Prec@1 95.703 (95.703)
 * Test Acc 95.000
------------------------------------------------
Epoch: [174/200][0/4]   LR: 0.01        Loss 0.1997 (0.1997)    Train Acc 94.531
(94.531)
 * Train Acc 94.750
Test: [0/2]     Loss 0.1297 (0.1297)    Prec@1 94.922 (94.922)
 * Test Acc 95.000
------------------------------------------------
```

```
Epoch: [175/200][0/4]   LR: 0.01        Loss 0.2331 (0.2331)    Train Acc 92.188
(92.188)
 * Train Acc 94.625
Test: [0/2]     Loss 0.1436 (0.1436)    Prec@1 94.922 (94.922)
 * Test Acc 95.667
------------------------------------------------
Epoch: [176/200][0/4]   LR: 0.01        Loss 0.1773 (0.1773)    Train Acc 96.094
(96.094)
 * Train Acc 96.125
Test: [0/2]     Loss 0.1556 (0.1556)    Prec@1 92.969 (92.969)
 * Test Acc 93.333
------------------------------------------------
Epoch: [177/200][0/4]   LR: 0.01        Loss 0.2150 (0.2150)    Train Acc 90.625
(90.625)
 * Train Acc 90.250
Test: [0/2]     Loss 0.2355 (0.2355)    Prec@1 86.719 (86.719)
 * Test Acc 87.000
------------------------------------------------
Epoch: [178/200][0/4]   LR: 0.01        Loss 0.2322 (0.2322)    Train Acc 87.891
(87.891)
 * Train Acc 92.125
Test: [0/2]     Loss 0.1579 (0.1579)    Prec@1 93.359 (93.359)
 * Test Acc 92.667
------------------------------------------------
Epoch: [179/200][0/4]   LR: 0.01        Loss 0.2478 (0.2478)    Train Acc 90.625
(90.625)
 * Train Acc 91.375
Test: [0/2]     Loss 0.1900 (0.1900)    Prec@1 91.406 (91.406)
 * Test Acc 90.333
------------------------------------------------
Epoch: [180/200][0/4]   LR: 0.01        Loss 0.2212 (0.2212)    Train Acc 91.797
(91.797)
 * Train Acc 92.375
Test: [0/2]     Loss 0.1810 (0.1810)    Prec@1 91.406 (91.406)
 * Test Acc 92.000
------------------------------------------------
```

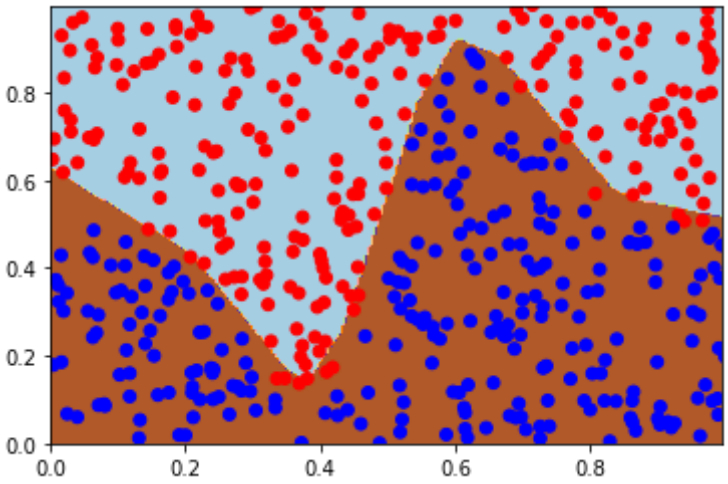



```
Epoch: [181/200][0/4]   LR: 0.01        Loss 0.2386 (0.2386)    Train Acc 88.281
(88.281)
 * Train Acc 91.125
Test: [0/2]     Loss 0.2499 (0.2499)    Prec@1 86.719 (86.719)
 * Test Acc 87.000
------------------------------------------------
Epoch: [182/200][0/4]   LR: 0.01        Loss 0.2768 (0.2768)    Train Acc 87.109
(87.109)
 * Train Acc 91.125
Test: [0/2]     Loss 0.1734 (0.1734)    Prec@1 92.188 (92.188)
 * Test Acc 92.333
------------------------------------------------
```

```
Epoch: [183/200][0/4]   LR: 0.01        Loss 0.2380 (0.2380)    Train Acc 91.016
(91.016)
 * Train Acc 90.250
Test: [0/2]     Loss 0.1723 (0.1723)    Prec@1 94.141 (94.141)
 * Test Acc 93.667
-----------------------------------------------
Epoch: [184/200][0/4]   LR: 0.01        Loss 0.2219 (0.2219)    Train Acc 92.578
(92.578)
 * Train Acc 94.625
Test: [0/2]     Loss 0.1438 (0.1438)    Prec@1 94.531 (94.531)
 * Test Acc 93.667
-----------------------------------------------
Epoch: [185/200][0/4]   LR: 0.01        Loss 0.1809 (0.1809)    Train Acc 92.969
(92.969)
 * Train Acc 91.125
Test: [0/2]     Loss 0.1968 (0.1968)    Prec@1 89.062 (89.062)
 * Test Acc 89.000
-----------------------------------------------
Epoch: [186/200][0/4]   LR: 0.01        Loss 0.1632 (0.1632)    Train Acc 95.312
(95.312)
 * Train Acc 91.500
Test: [0/2]     Loss 0.1332 (0.1332)    Prec@1 93.359 (93.359)
 * Test Acc 94.000
-----------------------------------------------
Epoch: [187/200][0/4]   LR: 0.01        Loss 0.1558 (0.1558)    Train Acc 94.141
(94.141)
 * Train Acc 91.125
Test: [0/2]     Loss 0.1447 (0.1447)    Prec@1 96.875 (96.875)
 * Test Acc 97.000
-----------------------------------------------
Epoch: [188/200][0/4]   LR: 0.01        Loss 0.1813 (0.1813)    Train Acc 93.359
(93.359)
 * Train Acc 92.125
Test: [0/2]     Loss 0.1243 (0.1243)    Prec@1 96.875 (96.875)
 * Test Acc 97.000
-----------------------------------------------
Epoch: [189/200][0/4]   LR: 0.01        Loss 0.1875 (0.1875)    Train Acc 95.703
(95.703)
 * Train Acc 95.375
Test: [0/2]     Loss 0.1269 (0.1269)    Prec@1 95.703 (95.703)
 * Test Acc 95.667
-----------------------------------------------
Epoch: [190/200][0/4]   LR: 0.01        Loss 0.1830 (0.1830)    Train Acc 95.312
(95.312)
 * Train Acc 95.375
Test: [0/2]     Loss 0.1424 (0.1424)    Prec@1 94.922 (94.922)
 * Test Acc 94.667
-----------------------------------------------
```



```
Epoch: [191/200][0/4]   LR: 0.01        Loss 0.1517 (0.1517)    Train Acc 96.094
(96.094)
 * Train Acc 96.250
Test: [0/2]     Loss 0.1312 (0.1312)    Prec@1 94.531 (94.531)
 * Test Acc 94.667
------------------------------------------------
Epoch: [192/200][0/4]   LR: 0.01        Loss 0.1806 (0.1806)    Train Acc 93.750
(93.750)
 * Train Acc 94.500
Test: [0/2]     Loss 0.1349 (0.1349)    Prec@1 96.094 (96.094)
 * Test Acc 96.000
------------------------------------------------
Epoch: [193/200][0/4]   LR: 0.01        Loss 0.1529 (0.1529)    Train Acc 95.703
(95.703)
 * Train Acc 94.750
Test: [0/2]     Loss 0.1190 (0.1190)    Prec@1 96.094 (96.094)
 * Test Acc 95.667
------------------------------------------------
Epoch: [194/200][0/4]   LR: 0.01        Loss 0.1695 (0.1695)    Train Acc 94.922
(94.922)
 * Train Acc 95.750
Test: [0/2]     Loss 0.1286 (0.1286)    Prec@1 96.875 (96.875)
 * Test Acc 97.000
------------------------------------------------
Epoch: [195/200][0/4]   LR: 0.01        Loss 0.1643 (0.1643)    Train Acc 97.266
(97.266)
 * Train Acc 95.375
Test: [0/2]     Loss 0.1247 (0.1247)    Prec@1 96.094 (96.094)
 * Test Acc 96.333
------------------------------------------------
Epoch: [196/200][0/4]   LR: 0.01        Loss 0.1971 (0.1971)    Train Acc 95.312
(95.312)
 * Train Acc 96.000
Test: [0/2]     Loss 0.0984 (0.0984)    Prec@1 96.875 (96.875)
 * Test Acc 96.000
------------------------------------------------
Epoch: [197/200][0/4]   LR: 0.01        Loss 0.2096 (0.2096)    Train Acc 93.750
(93.750)
 * Train Acc 94.875
Test: [0/2]     Loss 0.1272 (0.1272)    Prec@1 96.094 (96.094)
 * Test Acc 96.333
------------------------------------------------
Epoch: [198/200][0/4]   LR: 0.01        Loss 0.1903 (0.1903)    Train Acc 96.484
(96.484)
 * Train Acc 96.125
Test: [0/2]     Loss 0.1065 (0.1065)    Prec@1 96.094 (96.094)
 * Test Acc 95.333
------------------------------------------------
Epoch: [199/200][0/4]   LR: 0.01        Loss 0.1803 (0.1803)    Train Acc 94.531
(94.531)
 * Train Acc 95.000
Test: [0/2]     Loss 0.1534 (0.1534)    Prec@1 94.531 (94.531)
 * Test Acc 94.667
------------------------------------------------
```



(a) Plot how the training loss changes with epochs. Also, plot the training accuracy and validation accuracy in a single plot. Comment on the trends.

In [12]:

```
def train_new(train_loader, model, criterion, optimizer, epoch):
    batch_time = AverageMeter()
    data_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()

    # switch to train mode
    model.train()

    end = time.time()
    for i, (input, target) in enumerate(train_loader):
        # measure data loading time
        data_time.update(time.time() - end)

        target = target.to(device)
        input_var = torch.autograd.Variable(input).to(device)
        target_var = torch.autograd.Variable(target).to(device)
        # target_var = torch.squeeze(target_var)
        # compute output
        output = model(input_var)

        # compute loss
        loss = criterion(output, target_var.long())
        #print(loss)
        # measure accuracy and record loss
        prec1 = accuracy(output.data, target)
        losses.update(loss.item(), input.size(0))
        top1.update(prec1[0][0], input.size(0))

        # compute gradient and do SGD step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

    return losses.avg, top1.avg

def validate_accuracy(val_loader, model, criterion):
    batch_time = AverageMeter()
    losses = AverageMeter()
    top1 = AverageMeter()

    # switch to evaluate mode
    model.eval()

    end = time.time()
    for i, (input, target) in enumerate(val_loader):
        target = target.to(device)
        input_var = torch.autograd.Variable(input, volatile=True).to(device)
        target_var = torch.autograd.Variable(target, volatile=True).to(device)

        # compute output
        output = model(input_var)
        # loss = criterion(output, target_var[:,None])
        loss = criterion(output, target_var.long())

        # measure accuracy and record loss
        prec1 = accuracy(output.data, target)
        losses.update(loss.item(), input.size(0))
        top1.update(prec1[0][0], input.size(0))

        # measure elapsed time
        batch_time.update(time.time() - end)
        end = time.time()

    return top1.avg
```

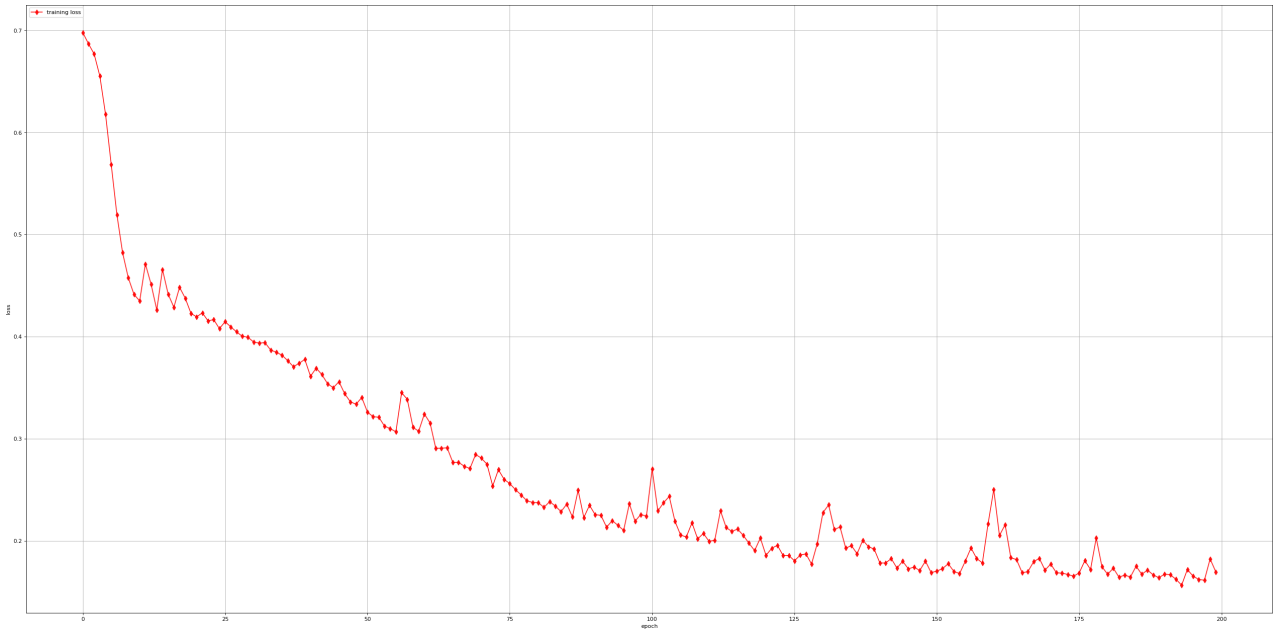
In [13]:

```
loss_list = []
acc_list = []
validation_acc_list = []

torch.manual_seed(999)
criterion = nn.CrossEntropyLoss().to(device)
model_1 = linear_nn(num_neurons,activations).to(device)
optimizer_1 = torch.optim.Adam(model_1.parameters(),lr=lr,weight_decay=weight_de

for e in range(1, 201):

    train_l, train_a = train_new(train_loader, model_1, criterion, optimizer_1,
    val_acc = validate_accuracy(val_loader, model_1, criterion)

    loss_list.append(train_l)
    acc_list.append(train_a)
    validation_acc_list.append(val_acc)
```

In [14]:

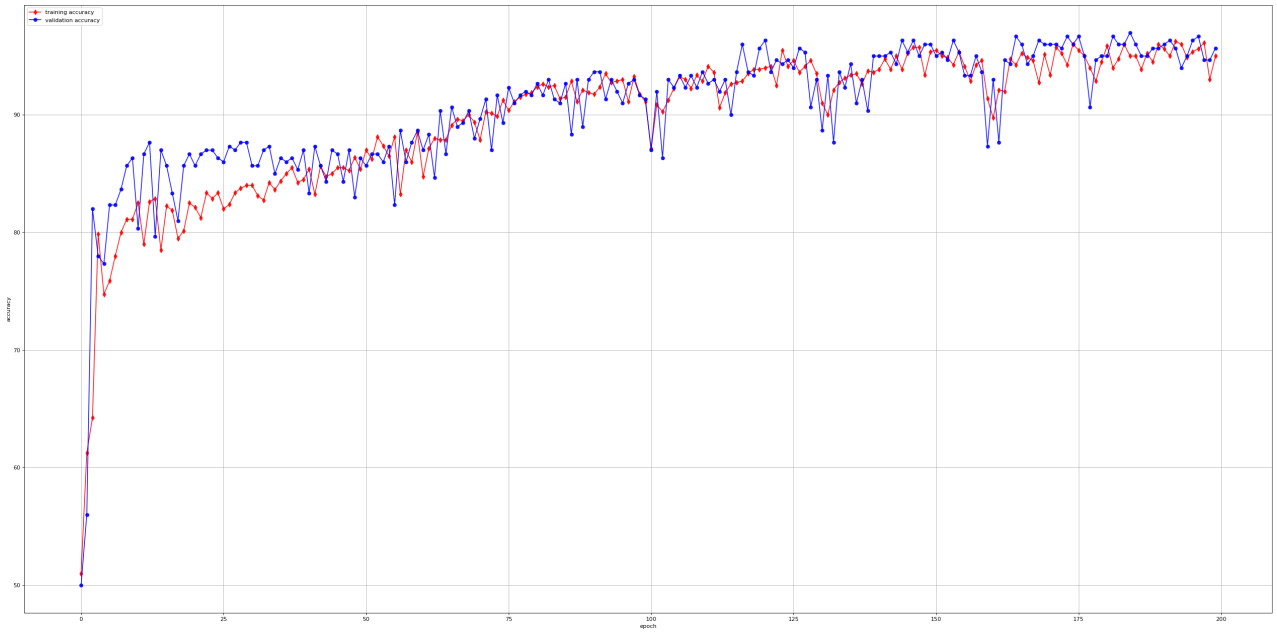
```
# plot training loss

figure(figsize = (40, 20), dpi = 80)
x = np.arange(200)
```

```
y = loss_list

plt.plot(x, y, c = 'r', label = 'training loss', marker = 'd', alpha = 0.9)
plt.xlabel("epoch")
plt.ylabel("loss")
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



In [15]:

```
# plot training accuracy and validation accuracy

figure(figsize = (40, 20), dpi = 80)
x = np.arange(200)
y1 = acc_list
y2 = validation_acc_list

plt.plot(x, y1, c = 'r', label = 'training accuracy', marker = 'd', alpha = 0.9)
plt.plot(x, y2, c = 'b', label = 'validation accuracy', marker = 'o', alpha = 0.
plt.xlabel("epoch")
plt.ylabel("accuracy")
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



Comment on the trends

1. From the training loss plot, we can see that the loss trend is getting smaller and smaller gradually.
2. From the training accuracy plot and validation accuracy plot, we can see that both of their accuracy is getting bigger and bigger until converged to a high number. the training accuracy and the validation accuracy are around similar values.

(b) Effect of learning rate: run the code with lr = [1, 0.1,0.01,0.001,0.0001]. Plot the training and validation accuracy for each of these learning rates. Comment on which learning rate is best and what is the issue with other learning rates.

In [16]:
```
torch.manual_seed(999)
criterion = nn.CrossEntropyLoss().to(device)
```

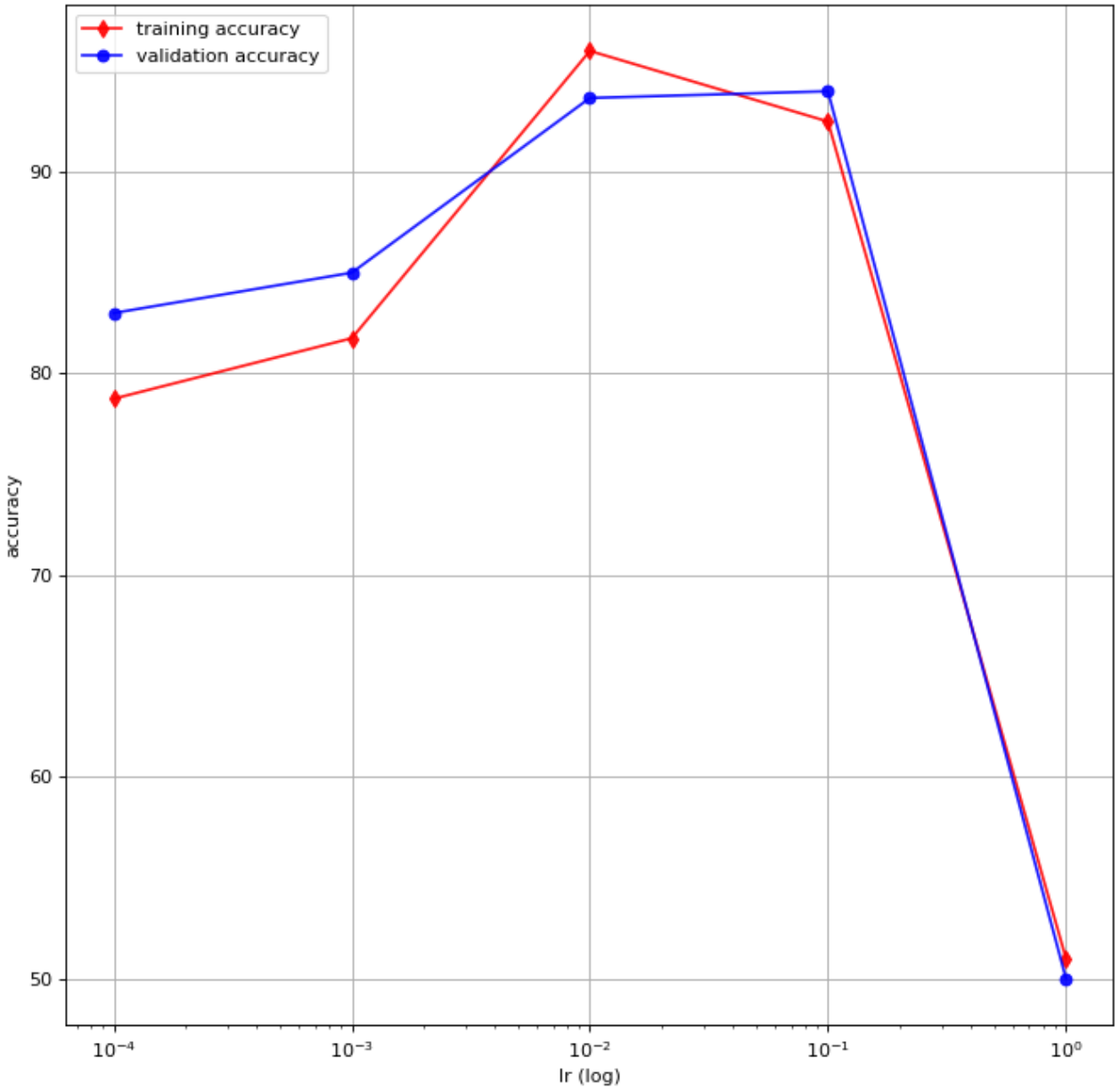
In [17]:
```
lr_list = [1, 0.1, 0.01, 0.001, 0.0001]
train_acc_list_lr = []
validation_acc_list_lr = []
acc_1 = []
acc_2 = []

for lr in lr_list:
    model_new = linear_nn(num_neurons,activations).to(device)
    optimizer_new = torch.optim.Adam(model_new.parameters(),lr=lr,weight_decay=w

    for e in np.arange(num_epochs):
        los, acc = train_new(train_loader, model_new, criterion, optimizer_new,
        acc_val = validate_accuracy(val_loader, model_new, criterion)
```

```
        acc_1.append(acc)
        acc_2.append(acc_val)


    train_acc_list_lr.append(acc)
    validation_acc_list_lr.append(acc_val)

print(train_acc_list_lr)
print(validation_acc_list_lr)
#print(acc_1)
#print(acc_2)
```

```
[tensor(51.), tensor(92.5000), tensor(96.), tensor(81.7500), tensor(78.7500)]
[tensor(50.), tensor(94.), tensor(93.6667), tensor(85.), tensor(83.)]
```

In [18]:

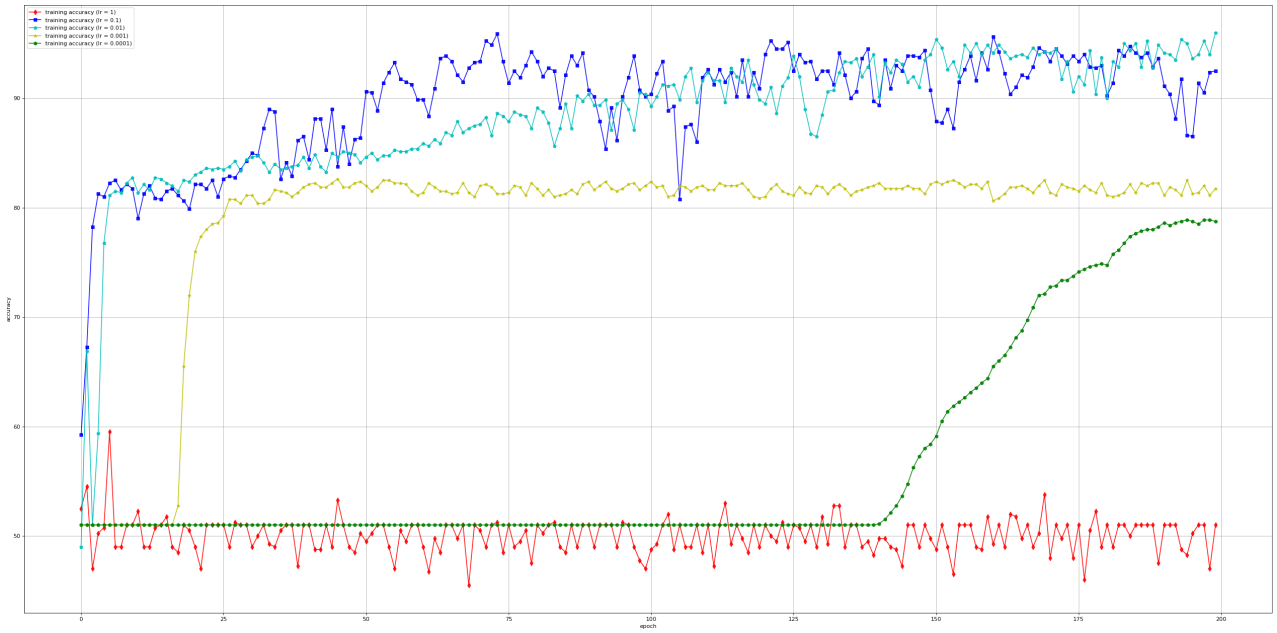
```
# plot training accuracy and validation accuracy after 200 epochs regarding diff

figure(figsize = (10, 10), dpi = 80)
y1 = train_acc_list_lr
y2 = validation_acc_list_lr

plt.plot(lr_list, y1, c = 'r', label = 'training accuracy', marker = 'd', alpha
plt.plot(lr_list, y2, c = 'b', label = 'validation accuracy', marker = 'o', alph
plt.xlabel("lr (log)")
plt.xscale('log')
plt.ylabel("accuracy")
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



In [19]:

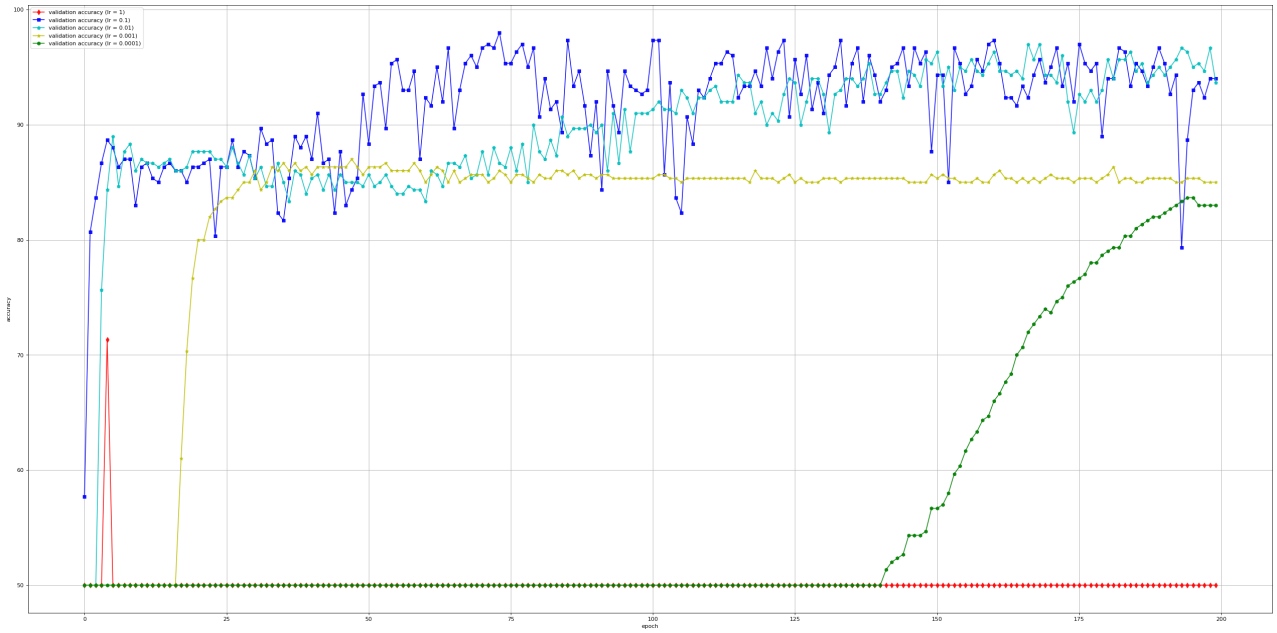
```
# plot training accuracy

figure(figsize = (40, 20), dpi = 80)
x = np.arange(200)
y1 = acc_1[0:200]
y2 = acc_1[200:400]
y3 = acc_1[400:600]
y4 = acc_1[600:800]
y5 = acc_1[800:1000]

plt.plot(x, y1, c = 'r', label = 'training accuracy (lr = 1)', marker = 'd', alp
plt.plot(x, y2, c = 'b', label = 'training accuracy (lr = 0.1)', marker = 's', a
plt.plot(x, y3, c = 'c', label = 'training accuracy (lr = 0.01)', marker = 'p',
plt.plot(x, y4, c = 'y', label = 'training accuracy (lr = 0.001)', marker = '*',
plt.plot(x, y5, c = 'g', label = 'training accuracy (lr = 0.0001)', marker = 'h'

plt.xlabel("epoch")
plt.ylabel("accuracy")
plt.legend(loc='upper left')
```

```
plt.grid()
plt.show()
```



In [20]:

```
# plot validation accuracy

figure(figsize = (40, 20), dpi = 80)
x = np.arange(200)
y1 = acc_2[0:200]
y2 = acc_2[200:400]
y3 = acc_2[400:600]
y4 = acc_2[600:800]
y5 = acc_2[800:1000]

plt.plot(x, y1, c = 'r', label = 'validation accuracy (lr = 1)', marker = 'd', a
plt.plot(x, y2, c = 'b', label = 'validation accuracy (lr = 0.1)', marker = 's',
plt.plot(x, y3, c = 'c', label = 'validation accuracy (lr = 0.01)', marker = 'p'
plt.plot(x, y4, c = 'y', label = 'validation accuracy (lr = 0.001)', marker = '*
plt.plot(x, y5, c = 'g', label = 'validation accuracy (lr = 0.0001)', marker = '

plt.xlabel("epoch")
plt.ylabel("accuracy")
plt.legend(loc='upper left')
plt.grid()
plt.show()
```



The learning rate lr = 0.01 is the best. The other learning rates are too small or too big.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

(c) The current model uses 3 hidden layers with 20, 10, and 10 neurons in the first, second, and third hidden layers respectively. Now, keeping the remaining parameters the same, change this to a model with 1 hidden layer containing 100 neurons. Report the final training and testing accuracies for both the models. Also, report the number of network parameters for these models. Comment on which model is the best. Is the deep network better or the shallow one?

In [21]:

```
lr = 0.01
torch.manual_seed(999)
criterion = nn.CrossEntropyLoss().to(device)

num_neurons_1 = [2,20,10,10,2] # list of neurons in each layer of NN.
num_neurons_2 = [2,100,2] # list of neurons in each layer of NN.
```

```
model_1 = linear_nn(num_neurons_1,activations=['relu'])
model_2 = linear_nn(num_neurons_2,activations=['relu'])

optimizer_new_1 = torch.optim.Adam(model_1.parameters(),lr=lr,weight_decay=weigh
optimizer_new_2 = torch.optim.Adam(model_2.parameters(),lr=lr,weight_decay=weigh
```

In [22]:

```
# Report the final training and testing accuracies for both the models
for e in np.arange(num_epochs):
    current_train_acc = train_new(train_loader, model_1, criterion, optimizer_ne
    current_validate_acc = validate_accuracy(val_loader, model, criterion)

    train_acc = train_new(train_loader, model_2, criterion, optimizer_new_2, epo
    validate_acc = validate_accuracy(val_loader, model, criterion)

print('The final training accuracy for current model:', current_train_acc)
print('The final testing accuracy for current model:', current_validate_acc)
print('The final training accuracy for new model:', train_acc)
print('The final testing accuracy for new model:', validate_acc)
```

```
The final training accuracy for current model: tensor(94.5000)
The final testing accuracy for current model: tensor(94.6667)
The final training accuracy for new model: tensor(93.)
The final testing accuracy for new model: tensor(94.6667)
```

In [23]:

```
# report the number of network parameters for these models
print('Number of network parameters for current model:', len(list(model_1.parame
print('Number of network parameters for new model:', len(list(model_2.parameters
```

```
Number of network parameters for current model: 8
Number of network parameters for new model: 4
```

The current model is better than the new model. The deep network is better.

In [ ]: