Machine Learning **Shreshth Tuli**
COL774 - Lab Assignment 1 2016CS10680
Supervisor - Prof. Parag Singla Date: 12/02/2019

Introduction

In this assignment, we implemented Simple Supervised Machine Learning and Deep Learning algorithms on Python. The algorithms and models were tested using real-life experimental data for accuracy and efficacy. We present different models, design decisions, parameters, hyper parameter tuning strategies used.
Libraries used:

- scikitlearn
- pytorch
- torchvision
- svmutil
- glob
- skimage
- pandas

SVM and PCA

Preprocessing and modeling:

- Took Grayscale cropped (144 x 144) images of sampled training data
- Reduced them to 50 principal components
- Applied binary gaussian SVM model for classification to positive class (reward at t+1 = 1) and negative class (reward at t+1 = 0)

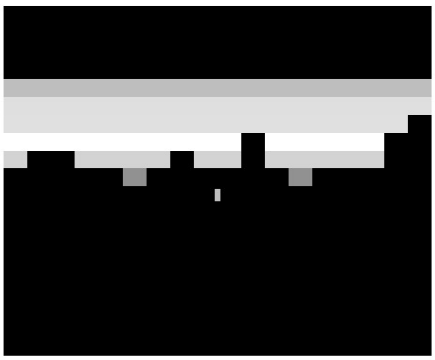

Figure 1: Grayscale cropped image

Validation for different g and C values:

- Best values of g and C : 0.1 and 1

g	c	Training accuracy	Validation accuracy
0.05	1	0.9496152436790033	0.517
0.1	1	0.9813810384204812	0.951
1	1	0.8273012938412090	0.692
0.1	0.1	0.9412049102947524	0.791
0.1	10	0.9781238714274918	0.912

Table 1: SVM accuracies

- If gamma is too large, the radius of the area of influence of the support vectors only includes the support vector itself and no amount of regularization with C will be able to prevent overfitting. When gamma is very small, the model is too constrained and cannot capture the complexity or "shape" of the data. The region of influence of any selected support vector would include the whole training set. The resulting model will behave similarly to a linear model with a set of hyperplanes that separate the centers of high density of any pair of two classes. For intermediate values, we can see on the second plot that good models can be found on a diagonal of C and gamma. Smooth models (lower gamma values) can be made more complex by increasing the importance of classifying each point correctly (larger C values) hence the diagonal of good performing models.

Best g and C = 0.1 and 1 respectively.

Convolutional Neural network

Single layer neural network with RGB images and negative classes down sampled to 5%.
Preprocessing (result shown in Fig-2):

- Cropped image to 144x144
- Converted to tensor

Tests:

- Tested for single layer neural network with hidden layer having different number of fully connected nodes and parameters like number of kernel in layers. All training done on same datataset and for 80 epochs.
- Best accuracy for : FC nodes = 4096 and Kernels = 32,64. If we increase kernels the model overfits the training data and doesn't generalize.

Number of FC units	Kernels	Training accuracy	Validation Accuracy
2048	32,64	96.128	93.913
2048	32,128	97.841	94.180
2048	64,64	97.910	94.719
4096	32,64	98.581	95.194
4096	48,64	99.002	94.401

Table 2: Accuracies for different cases - CNN

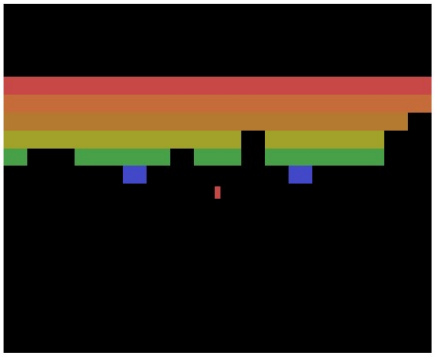

Figure 2: RGB cropped image

Competition Network

Model used = Resnet34. Other parameters:

1. Learning rate = 0.00005
2. Bath size = 1
3. Number of epochs on sampled dataset viz 100% positive samples and 2% negative samples = 40
4. Loss = CrossEntropyLoss
5. Optimizer = Adam

Google drive link: https://drive.google.com/open?id=1ErQT9okkNZ1TQCFKUcquGVI2xeV4MSnB.

----------	----------	----------
Layer (type)	Output Shape	Param #
==========	==========	==========
Conv2d-1	[-1, 64, 72, 72]	47,104
BatchNorm2d-2	[-1, 64, 72, 72]	128
ReLU-3	[-1, 64, 72, 72]	0
MaxPool2d-4	[-1, 64, 36, 36]	0
Conv2d-5	[-1, 64, 36, 36]	36,864
BatchNorm2d-6	[-1, 64, 36, 36]	128
ReLU-7	[-1, 64, 36, 36]	0
Conv2d-8	[-1, 64, 36, 36]	36,864
BatchNorm2d-9	[-1, 64, 36, 36]	128
ReLU-10	[-1, 64, 36, 36]	0
BasicBlock-11	[-1, 64, 36, 36]	0
Conv2d-12	[-1, 64, 36, 36]	36,864
BatchNorm2d-13	[-1, 64, 36, 36]	128
ReLU-14	[-1, 64, 36, 36]	0
Conv2d-15	[-1, 64, 36, 36]	36,864
BatchNorm2d-16	[-1, 64, 36, 36]	128
ReLU-17	[-1, 64, 36, 36]	0
BasicBlock-18	[-1, 64, 36, 36]	0
Conv2d-19	[-1, 64, 36, 36]	36,864
BatchNorm2d-20	[-1, 64, 36, 36]	128
ReLU-21	[-1, 64, 36, 36]	0
Conv2d-22	[-1, 64, 36, 36]	36,864
BatchNorm2d-23	[-1, 64, 36, 36]	128
ReLU-24	[-1, 64, 36, 36]	0
BasicBlock-25	[-1, 64, 36, 36]	0
Conv2d-26	[-1, 128, 18, 18]	73,728
BatchNorm2d-27	[-1, 128, 18, 18]	256
ReLU-28	[-1, 128, 18, 18]	0
Conv2d-29	[-1, 128, 18, 18]	147,456
BatchNorm2d-30	[-1, 128, 18, 18]	256
Conv2d-31	[-1, 128, 18, 18]	8,192
BatchNorm2d-32	[-1, 128, 18, 18]	256
ReLU-33	[-1, 128, 18, 18]	0
BasicBlock-34	[-1, 128, 18, 18]	0
Conv2d-35	[-1, 128, 18, 18]	147,456
BatchNorm2d-36	[-1, 128, 18, 18]	256
ReLU-37	[-1, 128, 18, 18]	0
Conv2d-38	[-1, 128, 18, 18]	147,456
BatchNorm2d-39	[-1, 128, 18, 18]	256
ReLU-40	[-1, 128, 18, 18]	0
BasicBlock-41	[-1, 128, 18, 18]	0
Conv2d-42	[-1, 128, 18, 18]	147,456
BatchNorm2d-43	[-1, 128, 18, 18]	256
ReLU-44	[-1, 128, 18, 18]	0
Conv2d-45	[-1, 128, 18, 18]	147,456
BatchNorm2d-46	[-1, 128, 18, 18]	256
ReLU-47	[-1, 128, 18, 18]	0
BasicBlock-48	[-1, 128, 18, 18]	0
Conv2d-49	[-1, 128, 18, 18]	147,456
BatchNorm2d-50	[-1, 128, 18, 18]	256
ReLU-51	[-1, 128, 18, 18]	0
Conv2d-52	[-1, 128, 18, 18]	147,456
BatchNorm2d-53	[-1, 128, 18, 18]	256
ReLU-54	[-1, 128, 18, 18]	0
BasicBlock-55	[-1, 128, 18, 18]	0
Conv2d-56	[-1, 256, 9, 9]	294,912
BatchNorm2d-57	[-1, 256, 9, 9]	512
ReLU-58	[-1, 256, 9, 9]	0
Conv2d-59	[-1, 256, 9, 9]	589,824
BatchNorm2d-60	[-1, 256, 9, 9]	512

Table 3:4Caption

----------	----------	----------
Layer (type)	Output Shape	Param #
==========	==========	==========
Conv2d-61	[-1, 256, 9, 9]	32,768
BatchNorm2d-62	[-1, 256, 9, 9]	512
ReLU-63	[-1, 256, 9, 9]	0
BasicBlock-64	[-1, 256, 9, 9]	0
Conv2d-65	[-1, 256, 9, 9]	589,824
BatchNorm2d-66	[-1, 256, 9, 9]	512
ReLU-67	[-1, 256, 9, 9]	0
Conv2d-68	[-1, 256, 9, 9]	589,824
BatchNorm2d-69	[-1, 256, 9, 9]	512
ReLU-70	[-1, 256, 9, 9]	0
BasicBlock-71	[-1, 256, 9, 9]	0
Conv2d-72	[-1, 256, 9, 9]	589,824
BatchNorm2d-73	[-1, 256, 9, 9]	512
ReLU-74	[-1, 256, 9, 9]	0
Conv2d-75	[-1, 256, 9, 9]	589,824
BatchNorm2d-76	[-1, 256, 9, 9]	512
ReLU-77	[-1, 256, 9, 9]	0
BasicBlock-78	[-1, 256, 9, 9]	0
Conv2d-79	[-1, 256, 9, 9]	589,824
BatchNorm2d-80	[-1, 256, 9, 9]	512
ReLU-81	[-1, 256, 9, 9]	0
Conv2d-82	[-1, 256, 9, 9]	589,824
BatchNorm2d-83	[-1, 256, 9, 9]	512
ReLU-84	[-1, 256, 9, 9]	0
BasicBlock-85	[-1, 256, 9, 9]	0
Conv2d-86	[-1, 256, 9, 9]	589,824
BatchNorm2d-87	[-1, 256, 9, 9]	512
ReLU-88	[-1, 256, 9, 9]	0
Conv2d-89	[-1, 256, 9, 9]	589,824
BatchNorm2d-90	[-1, 256, 9, 9]	512
ReLU-91	[-1, 256, 9, 9]	0
BasicBlock-92	[-1, 256, 9, 9]	0
Conv2d-93	[-1, 256, 9, 9]	589,824
BatchNorm2d-94	[-1, 256, 9, 9]	512
ReLU-95	[-1, 256, 9, 9]	0
Conv2d-96	[-1, 256, 9, 9]	589,824
BatchNorm2d-97	[-1, 256, 9, 9]	512
ReLU-98	[-1, 256, 9, 9]	0
BasicBlock-99	[-1, 256, 9, 9]	0
Conv2d-100	[-1, 512, 5, 5]	1,179,648
BatchNorm2d-101	[-1, 512, 5, 5]	1,024
ReLU-102	[-1, 512, 5, 5]	0
Conv2d-103	[-1, 512, 5, 5]	2,359,296
BatchNorm2d-104	[-1, 512, 5, 5]	1,024
Conv2d-105	[-1, 512, 5, 5]	131,072
BatchNorm2d-106	[-1, 512, 5, 5]	1,024
ReLU-107	[-1, 512, 5, 5]	0
BasicBlock-108	[-1, 512, 5, 5]	0
Conv2d-109	[-1, 512, 5, 5]	2,359,296
BatchNorm2d-110	[-1, 512, 5, 5]	1,024
ReLU-111	[-1, 512, 5, 5]	0
Conv2d-112	[-1, 512, 5, 5]	2,359,296
BatchNorm2d-113	[-1, 512, 5, 5]	1,024
ReLU-114	[-1, 512, 5, 5]	0
BasicBlock-115	[-1, 512, 5, 5]	0
Conv2d-116	[-1, 512, 5, 5]	2,359,296
BatchNorm2d-117	[-1, 512, 5, 5]	1,024
ReLU-118	[-1, 512, 5, 5]	0
Conv2d-119	[-1, 512, 5, 5]	2,359,296
BatchNorm2d-120	[-1, 512, 5, 5]	1,024

Table 4:5Caption

----------	----------	----------
Layer (type)	Output Shape	Param #
==========	==========	==========
ReLU-121	[-1, 512, 5, 5]	0
BasicBlock-122	[-1, 512, 5, 5]	0
AdaptiveAvgPool2d-123	[-1, 512, 1, 1]	0
Linear-124	[-1, 2]	1,026
==========	==========	==========
Total params: 21,323,394		
Trainable params: 21,323,394		
Non-trainable params: 0		
----------	----------	----------
Input size (MB): 1.19		
Forward/backward pass size(MB): 40.22		
Params size (MB): 81.34		
Estimated Total Size (MB):122.75		
----------	----------	----------

Table 5: Resnet34 architectural details