ECS629U-759P: ARTIFICIAL INTELLIGENCE
2019/20 – Semester 2 (CW-2)
Name: Sushma Ghogale
Student ID: 190635387

Q1) (a)

Robin has a Dog	*∃x. Dog(x) ∧ Owns (Robin, x)*
Robin buys carrots by the bushel.	BUY(Robin) or BUY(x,y)
Anyone who owns a rabbit hates anything that chases any rabbit.	*∀x ∀y (OWNS(x,y) ∧ RABBIT(y) → ∀z ∀w (RABBIT(w) ∧ CHASE(z,w) → HATES(x,z)))*
Every dog chases some rabbit.	*∀x (DOG(x) → ∃y (RABBIT(y) ∧ CHASE(x,y)))*
Anyone who buys carrots by the bushel owns either a rabbit or a grocery store.	*∀x (BUY(x) → ∃y (OWNS(x,y) ∧ (RABBIT(y) ∨ GROCERY(y))))*
Someone who hates something owned by another person will not date that person.	*∀x ∀y ∀z (OWNS(y,z) ∧ HATES(x,z) → ¬ DATE(x,y))*

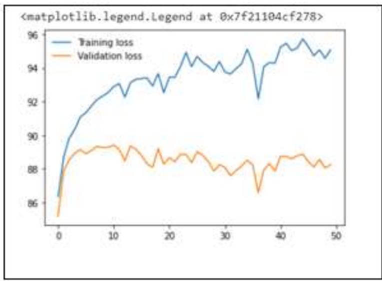
Q(1)(b)
¬ Dog(x) V Owns (Robin, x)
Buy(x,y) or BUY(Robin,x)
¬OWNS(x,y) V RABBIT(y) V ¬ (RABBIT(w) ∧ CHASE(z,w) V ¬ HATES(x,z)
¬BUY(x) v OWNS(x,y) v ¬ RABBIT(y) ∧ GROCERY(y)
OWNS(y,z) V HATES(x,z) V DATE(x,y)

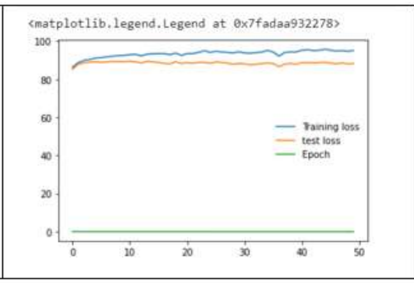
Q(1)(C) • If the person you are looking for does not own a grocery store, she will not date you.
If Mary does not own a grocery store, she will not date Robin.
((¬ ∃x (GROCERY(x) ∧ OWN(Mary,x))) → ¬ DATE(Mary,you))
CNF:
GROCERY(x) V ¬ OWN(Mary,x) V DATE(Mary,you)

Q(2)(a) Appropriate loss function to use us 'ADAM '- this stands for 'Adaptive Moment Estimator. This optimizer calculates different learning rate and works well in practice. It is faster compared to SGD and outperforms other techniques.

Q(2)(b)**ReLU as the activation function, a learning rate of 0.1 with the SGD optimiser.**

Epoch: 50 of 50, Train Acc: 95.068, Test Acc: 88.250, Loss: 0.010



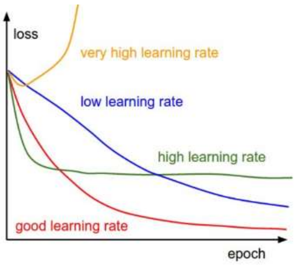


Test accuracy of the model on the 10000 test images : 88.25%

Train and test Accuracy gradually increases initially as the Loss decreases. After few epocs Training accuracy keeps fluctuating and increases, but test accuracy does not increase much as the loss gradually decreases and Loss will be almost zero when epoch reaches to 50.

Q(2)(c)
The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. Adaptive learning rates can

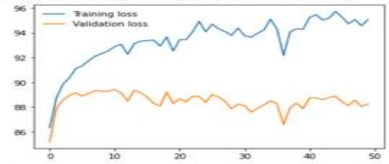
accelerate training. Smaller the Learning rate better the performance and train and test accuracy increase while decreasing the loss function. But higher Learning rate will not affect loss function i.e(LR=10) loss= none ref(https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10)



Good learning rate is the one which Loss gradually decreases w.r.to epocs.

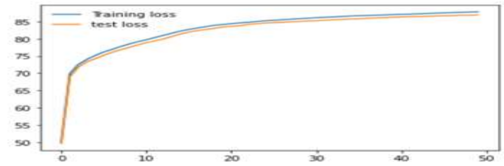
LEARNING_RATE = 0.1

<matplotlib.legend.Legend at 0x7f21104cf278>

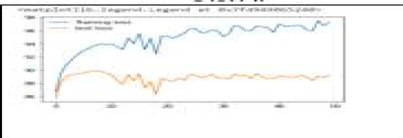
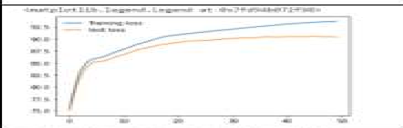
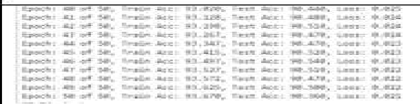
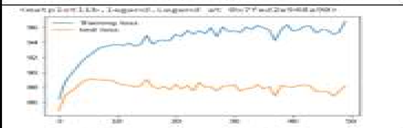
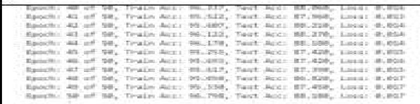
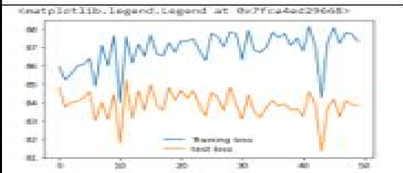
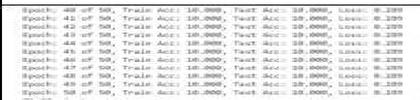


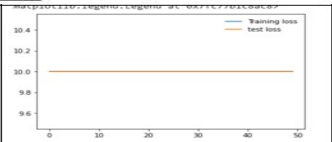
Test Accuracy of the model on the 10000 test images: 88.25 %

LEARNING_RATE=0.001

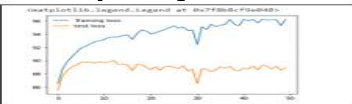
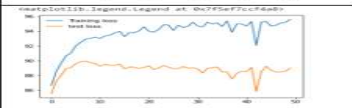
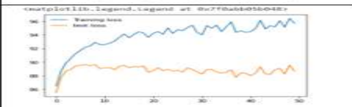


Epoch: 50 of 50, Train Acc; 87.853, Test Acc: 86.950, Loss: 0.044

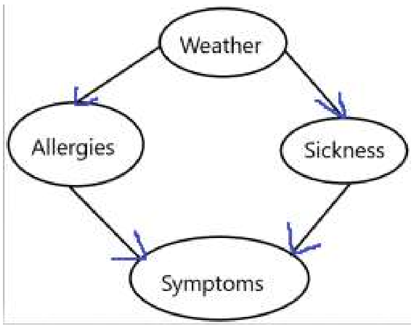
Activation = Tanh			Test Accuracy of the model on the 10000 test images: 89.2 %
Activation=Sigmoid			Test Accuracy of the model on the 10000 test images: 90.36%
Activation= Elu			
Learning rate=0.5		Epoch: 40 of 50, Train Acc: 87.513, Test Acc: 83.660, Loss: 0.044 Epoch: 41 of 50, Train Acc: 86.780, Test Acc: 83.240, Loss: 0.043 Epoch: 42 of 50, Train Acc: 88.120, Test Acc: 84.560, Loss: 0.044 Epoch: 43 of 50, Train Acc: 87.068, Test Acc: 83.890, Loss: 0.043 Epoch: 44 of 50, Train Acc: 84.250, Test Acc: 81.310, Loss: 0.044 Epoch: 45 of 50, Train Acc: 87.267, Test Acc: 83.670, Loss: 0.045 Epoch: 46 of 50, Train Acc: 88.072, Test Acc: 84.170, Loss: 0.047 Epoch: 47 of 50, Train Acc: 87.215, Test Acc: 83.210, Loss: 0.044 Epoch: 48 of 50, Train Acc: 87.798, Test Acc: 84.090, Loss: 0.047 Epoch: 49 of 50, Train Acc: 87.723, Test Acc: 83.860, Loss: 0.043 Epoch: 50 of 50, Train Acc: 87.320, Test Acc: 83.830, Loss: 0.047	
Learning rate= 1			

LEARNING_RATE =10		Epoch: 40 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 41 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 42 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 43 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 44 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 45 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 46 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 47 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 48 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 49 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan Epoch: 50 of 50, Train Acc: 10.000, Test Acc: 10.000, Loss: nan

Q(2)(d) Overfitting is the major problem in the large networks, which becomes slow and difficult to deal with overfitting. Dropout is a technique for addressing this issue. The key is to randomly drop units from Neural network during training. During training, dropout samples from an exponential number of different “thinned” networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization method.

Dropout=0.25		Test Accuracy of the model on the 10000 test images: 88.9 % Epoch: 50 of 50, Train Acc: 96.188, Test Acc: 88.900, Loss: 0.009
Dropout=0.4		Test Accuracy of the model on the 10000 test images: 88.99000000000001 Test Accuracy of the model on the 10000 test images: 88.99000000000001 %
Dropout=0.3		Epoch: 50 of 50, Train Acc: 95.713, Test Acc: 88.670, Loss: 0.010 Test Accuracy of the model on the 10000 test images: 88.67 %

Q(3)(a)



Q(3)(b)

Rainey	Sunny	Windy
0.25	0.6	0.15

Weather	Allergies
Rainey	0.15
Windy	0.6
Sunny	0.5

Weather	Allergy
Rainey	0.8
Windy	0.45
Sunny	0.15

Aller	Sick	P(s=headache)	P(s=Sunny)	P(s=stom)
True	True	0.6	0.3	0.1
True	False	0.75	0.2	0.05
False	True	0.1	0.5	0.4
False	False	0.15	0.05	0.8

Q(3)(C)

Si=Sickness, h=headaches, S=Sunny, A= Allur

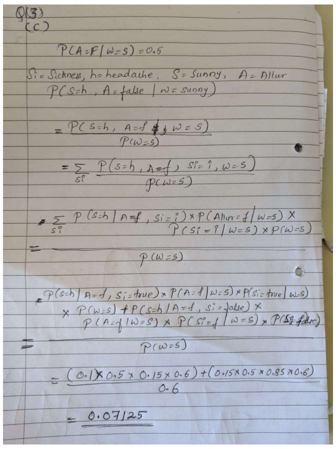
$$P(s=h, A=false \,/\, w = sunny) = \frac{P(s=h, a=f, w=s)}{P(w=s)}$$

$$= \frac{\sum_{Si} P(s=h, A=f, Si=i, w=s)}{P(w=s)}$$

$$= \frac{\sum_{Si} P(s=h \mid A=f, Si=i) * P(Allur=f \mid w=s) * P(Si=i \mid w=s) * P(w=s)}{P(w=s)}$$

$$= \frac{P(s=h|A=f,Si=true)*P(A=f|w=s)*P(Si=true/w=s)*P(w=s)+P(s=h|A=f,Si=false)*P(A=f|w=s)*P(Si=f|w=s)*P(Si=false)}{P(w=s)}$$

$$= \frac{(0.1*0.5*0.15*0.6)+(0.15*0.5*0.85*0.6)}{0.6} = \mathbf{0.007125}$$


Q(3)
(c)

P(A=F | w=s) = 0.5

Si = Sickness, h = headache, S = Sunny, A = Allur
P(S=h, A=false | w=sunny)

$= \frac{P(S=h,\ A=f,\ w=S)}{P(w=S)}$

$= \sum_{si} \frac{P(s=h,\ A=f,\ si=i,\ w=s)}{P(w=S)}$

$= \frac{\sum_{si} P(s=h \mid A=f,\ si=i) \times P(Allur=f \mid w=s) \times P(si=i \mid w=s) \times P(w=s)}{P(w=s)}$

$= \frac{P(s=h \mid A=f,\ si=true) \times P(A=f \mid w=s) \times P(si=true \mid w=s) \times P(w=s) + P(s=h \mid A=f,\ si=false) \times P(A=f \mid w=S) \times P(si=f \mid w=s) \times P(Si=false)}{P(w=s)}$

$= \frac{(0.1 \times 0.5 \times 0.15 \times 0.6) + (0.15 \times 0.5 \times 0.85 \times 0.6)}{0.6}$

= 0.07125