First part: Crystal clear (Logic problem)

1.

1• ∃x Dog(x) ∧ Own(You,x)
2• Buy(ROBIN)
3• ∀x (∃y(Own(x,y)∧Rabbit(y)) ⇒ (∀z (∃w (Rabbit(w)∧Chase(z,w)) ⇒ Hate(x,z))))
4•∀x Dog(x) ⇒ ∃y (Rabbit(y) ∧ Chase(x,y))
5• ∀x Buy(x) ⇒ ∃y (Own(x,y) ∧ (Rabbit(y) ∨ Grocery(y)))
6• ∀x ∀y (∃z(Own(y,z) ∧ Hate(x,z)) ⇒ ¬Date(x,y))

2.

1) Remove implications:
3. ∀x(¬∃y(Own(x,y) ∧ Rabbit(y)))∨(∀z(¬∃w(Rabbit(w) ∧ Chase(z,w)) ∨ Hate(x,z))))
4. ∀x¬Dog(x)∨∃y(Rabbit(y) ∧ Chase(x,y))
5. ∀x¬Buy(x) ∨ ∃y (Own(x,y) ∧ (Rabbit(y) ∨ Grocery(y)))
6. ∀x ∀y (¬∃z(Own(y,z) ∧ Hate(x,z))) ∨ ¬Date(x,y))

2) Minimise negations:
3. ∀x∀y(¬Own(x,y) ∨ ¬Rabbit(y)) ∨ (∀z∀w (¬Rabbit(w) ∨ ¬Chase(z,w)) ∨ Hate(x,z))
6. ∀x ∀y ∀z(¬Own(y,z) ∨ ¬Hate(x,z)) ∨ ¬Date(x,y)

3) Standardise variables apart:
3. ∀x1∀y1(¬Own(x1,y1) ∨ ¬Rabbit(y1)) ∨ (∀z1∀w1 (¬Rabbit(w1) ∨ ¬Chase(z1,w1)) ∨ Hate(x1,z1))
4. ∀x2¬Dog(x2)∨∃y2(Rabbit(y2) ∧ Chase(x2,y2))
5. ∀x3¬Buy(x3)∨∃y3(Own(x3,y3) ∧ (Rabbit(y3) ∨ Grocery(y3)))
6. ∀x4∀y4∀z2(¬Own(y4,z2)∨¬Hate(x4,z2)) ∨ ¬Date(x4,y4)

4) Skolemise existentials:
1.Dog(D) ∧ Own(YOU,D)
3. ∀x2¬Dog(x2)∨(Rabbit(R(x2)) ∧ Chase(x2,R(x2))
the existential on y2 was in the scope of the universal on x, hence the need for introduction of the function R(x2).
5. ∀x3¬Buy(x3) ∨ (Own(x3,F(x3)) ∧ (Rabbit(F(x3)) ∨ Grocery(F(x3))))
the existential on y3 was in the scope of the universal on x, hence the need for introduction of the function.

5) Drop universals:
3. ⬚¬Own(x1,y1) ∨ ¬Rabbit(y1) ∨ ⬚ ¬Rabbit(w1) ∨ ¬Chase(z1,w1) ∨ Hate(x1,z1)
4. ⬚ ¬Dog(x2) ∨ (Rabbit(R(x2)) ∧ Chase(x2,R(x2))
5. ⬚ ¬Buy(x3) ∨ (Own(x3,F(x3)) ∧ (Rabbit(F(x3)) ∨ Grocery(F(x3))))
6. ⬚¬Own(y4,z2)∨¬Hate(x4,z2) ∨ ¬Date(x4,y4)

6) Convert to CNF:
4.[¬Dog(x2) ∨ Rabbit(R(x2))]∧ [¬Dog(x2) ∨ Chase(x2,R(x2))]
5. [¬Buy(x3) ∨ Own(x3,F(x3))] ∧ [¬Buy(x3) ∨ (Rabbit(F(x3)) ∨ Grocery(F(x3)))]

Final set:
1. Dog(D) ∧ Own(You,D)
2. Buy(ROBIN)

3. ¬Own(x1,y1) ∨ ¬Rabbit(y1) ∨ ¬Rabbit(w1) ∨ ¬Chase(z1,w1) ∨ Hate(x1,z1)
4.[¬Dog(x2) ∨ Rabbit(R(x2))] ∧ [¬Dog(x2) ∨ Chase(x2,R(x2))]
5.[¬Buy(x3) ∨ Own(x3,F(x3))] ∧ [¬Buy(x3) ∨ (Rabbit(F(x3)) ∨ Grocery(F(x3)))]
6. ¬Own(y4,z2) ∨ ¬Hate(x4,z2) ∨ ¬Date(x4,y4)

3.
Goal: ¬∃y(Own(ROBIN, y)∧Grocery(y))⇒¬Date(ROBIN, YOU)

Negated goal: ¬(¬∃y(Own(ROBIN, y)∧Grocery(y))⇒¬Date(ROBIN, YOU))

1) Remove implications: ¬(¬ (¬∃y(Own(ROBIN, y)∧Grocery(y))) ∨ ¬Date(ROBIN, YOU))

2) Minimise negations:
De Morgan's Law:
¬(¬ (¬∃y(Own(ROBIN, y)∧Grocery(y)))) ∧ Date(ROBIN, YOU)
Double negations:
(¬∃y(Own(ROBIN, y)∧Grocery(y))) ∧ Date(ROBIN, YOU)
(∀y ¬(Own(ROBIN, y)∧Grocery(y))) ∧ Date(ROBIN, YOU)
De Morgan's law:
∀y(¬Own(ROBIN, y)∨¬Grocery(y)) ∧ Date(ROBIN, YOU)

3) Standardise variables apart: ∀y5 (¬Own(ROBIN, y5)∨¬Grocery(y5)) ∧ Date(ROBIN, YOU)

4) Skolemise existentials: ∀y5(¬Own(ROBIN, F(y5))∨¬Grocery(F(y5))) ∧ Date(ROBIN, YOU)

5) Drop universals: □(¬Own(ROBIN, F(y5))∨¬Grocery(F(y5))) ∧ Date(ROBIN, YOU)

6) Convert to CNF: Nothing to do

4.
7-8: Split into disjunctive clauses & standardise variables apart:
1. Dog(D)
2. Own(You, D)
3. Buy(ROBIN)
4. ¬Own(x1,y1) ∨ ¬Rabbit(y1) ∨ ¬Rabbit(w1) ∨ ¬Chase(z1,w1) ∨ Hate(x1,z1)
5.¬Dog(x2) ∨ Rabbit(R(x2))
6.¬Dog(x3) ∨ Chase(x3,R(x3))
7.¬Buy(x4) ∨ Own(x4,F(x4))
8.¬Buy(x5) ∨ (Rabbit(F(x5)) ∨ Grocery(F(x5)))
9.¬Own(y2,z2) ∨ ¬Hate(x7,z2) ∨ ¬Date(x7,y2)
10.¬Own(ROBIN, F(y5))∨¬Grocery(F(y5))
11. Date(ROBIN, YOU)

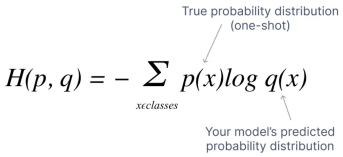
Resolution proof:
12. ¬Own(YOU,z2) ∨ ¬Hate(ROBIN,z2) (Unifier : ROBIN/x7, YOU/y2) < Resolve 9, 11>
13. ¬Hate(ROBIN,D) (Unfier D/z2) < Resolve 2 , 12>
14. ¬Own(ROBIN,y1) ∨ ¬Rabbit(y1) ∨ ¬Rabbit(w1) ∨ ¬Chase(D,w1) (Unifier ROBIN/x1, D/z1) < Resolve 4, 13>
15. Own(ROBIN,F(ROBIN)) (Unifier ROBIN/x4) < Resolve 3, 7>

16. ¬Rabbit(y1) ∨ ¬Rabbit(w1) ∨ ¬Chase(D,w1) (Unifier F(ROBIN)/y1) < Resolve 14, 15>
17. Chase(D,R(D)) (Unifier D/x3) < Resolve 1, 6>
18. ¬Rabbit(y1) ∨ ¬Rabbit(R(D)) (Unifier R(D)/w1) < Resolve 16,17>
19. Rabbit(R(D)) (Unifier D/x2) < Resolve 5, 1>
20. ¬Rabbit(y1) < Resolve 18, 19>
21. Rabbit(F(ROBIN)) ∨ Grocery(F(ROBIN)) (Unifier ROBIN /x5) < Resolve 3, 8>
22 . Grocery(F(ROBIN)) (Unifier F(ROBIN)/y1) < Resolve 20, 21>
23. ¬Own(ROBIN, F(ROBIN)) (Unifier F(ROBIN)/F(y5)) < Resolve 10, 22>
24. ⦰ < Resolve 15, 23>

Part two:

1.

We use cross-entropy loss since our problem is multi-class classification and this loss is suitable for it. Cross-entropy loss penalizes the model more when its prediction is wrong with high confidence. This loss is based on probabilities; when the probability of prediction for the correct class is low, the loss is high.



Also, Cross-entropy loss provides smooth gradients that lead to effective backpropagation in training.

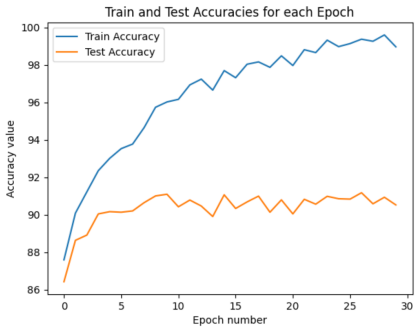
This formula measures the dissimilarity between p(x) and q(x) probability distributions (ground truth and predicted distributions respectively).

p(x) is the true probability(ground truth label) for x which is one-hot encoded (1 for its class and 0 for other classes)
q(x) is the predicted probability for x by the model.
We use the logarithm function to penalize the model more when it is wrong and confident (p(x) low and q(x) high). Then we sum over all x values and negate it.

2. (a)

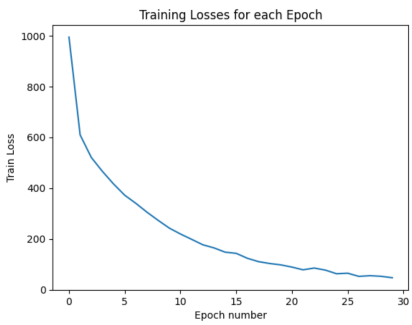
Last Train Accuracy: 98.96166666666667 Last Test Accuracy: 90.52

2. (b)



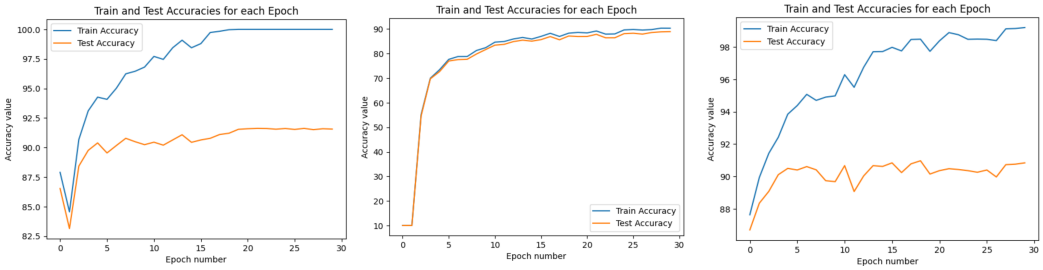
We can see that the training accuracy increases with high speed over epochs. Meanwhile, although the test accuracy increases over the first 10 epochs, it fluctuates and remains almost the same over the rest of the epochs (10 to 30). We may have overfitting to the training data since the training accuracy is considerably higher than the test accuracy means that our model captures even noises from training data and that is why it has excellent performance on the train set and its performance decreases noticeably on the test set.

2. (c)



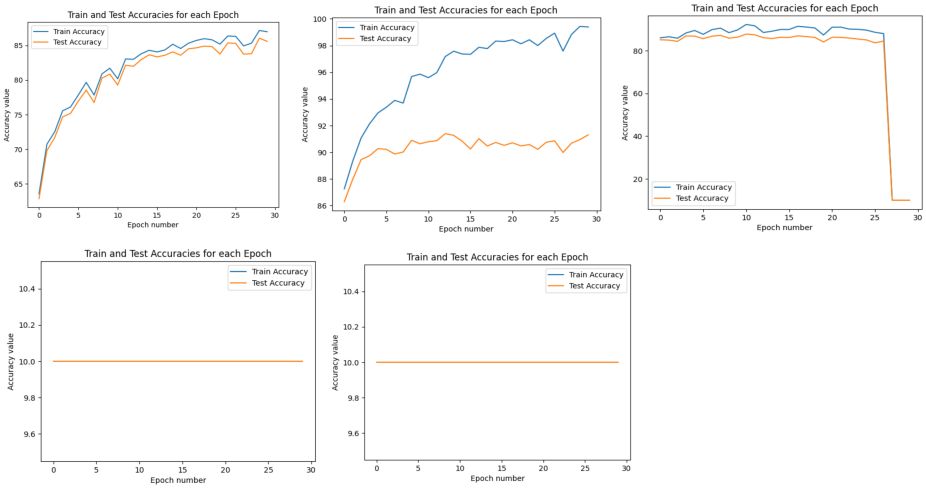
We can observe that the loss value per each epoch in the training process decreases sharply over the first 15 epochs. From epoch 15 to 30, the graph also has a reduction but with a lower speed.

3.



Tanh: Last Train Accuracy: 100.0 - Last Test Accuracy: 91.56
Sigmoid: Last Train Accuracy: 90.36 - Last Test Accuracy: 88.96
ELU: Last Train Accuracy: 99.19666666666667 - Last Test Accuracy: 90.84
We can see that the Tanh activation has the highest accuracy on the train and test set. ELU and Sigmoid are ranked next in terms of accuracy. From the graphs, we can observe that both train and test accuracies are high for the sigmoid activation function graph (second fig). However, there is a noticeable difference between the accuracies of the train and test set in the graphs of Tanh and ELU activations (first and third figs) which may show overfitting.

4.



ReLU lr 0.001:
Epoch 30: loss: 688.6943768635392, train accuracy: 86.96166666666667, test accuracy:85.58
ReLU lr 0.1:
Epoch 30: loss: 46.52681931552979, train accuracy: 99.39166666666667, test accuracy:91.3
ReLU lr 0.5:
Epoch 30: loss: 4323.820770025253, train accuracy: 10.0, test accuracy:10.0
ReLU lr 1:
Epoch 30: loss: 4331.122264385223, train accuracy: 10.0, test accuracy:10.0
ReLU lr 10:
Epoch 30: loss: nan, train accuracy: 10.0, test accuracy:10.0

We can observe that by increasing the learning rate from 0.001 to 0.1 the loss value decreases sharply meaning that we are closer to the minima by changing the step size and we see a noticeable increase in

accuracies. By increasing the learning rate to 0.5 we observe that the loss value increases significantly means that our step size is bigger so we probably pass the minima and that's why our accuracies drop suddenly in the last epochs (fig 3). When we change the step size to 1 and 10 we see that our loss value increases significantly or becomes Nan (exploding gradients) compared to other learning rates and our accuracy is too low and stable. This is because our step size is too big so we pass the minima and our loss wouldn't decrease as expected during the training process. Also, when the step size is too high, the weights of our model become so large leads to divergence of the optimization process rather than converge.
Trade-offs:
For small learning rates like 0.001, the risk of divergence is low and we have a stable situation but the convergence speed is slow and there is a danger of being stuck in the local minima. For moderate learning rates, we observe faster speed to converge and better accuracy while the risk of exploding gradients becomes a little bit higher than before. When we change the learning rate to high values
We converge so fast but we have a high risk of divergence since we may pass the minima and exploding gradients may happen.

5.

ReLU lr 0.1
dropout of 0.3 rate
Last train accuracy: 99.14833333333333, Last test accuracy:91.32
We use dropout to deal with the overfitting problem and avoid it. We can observe a slight increase in both training and testing accuracies.

This slight change may happen because of the following reasons:
1- the dropout rate is too low so the regularization effect is not noticeable.
2- if the model is not overfitting applying the dropout technique will not show a noticeable improvement.
3- if the dataset is too small the effect of the dropout technique is not considerable. This is because in this method we want to avoid the model memorizing the training dataset and let the model learn more robust features.
4- number of the epochs is low and the training process should continue more to see the dropout influence.