1. Crystal Clear

(a). FOL

1. ∃x (dog(x) ^ own(You, x))
2. ∃x buys_carrot(x) ^ looking_for(You, x)
3. ∀a ∃b owns(a, b) ^ rabbit(b) → ∀c ∀d hates(a, c) ^ chases(c, d)
4. ∀x dog(x) → ∃y rabbit(y) ^ chases(x, y)
5. ∀x buys_carrot(x) → ∃y owns(x, y) ^ {rabbit(y) v grocery(y)}
6. ∀x ∀y ∀z owns(x, y) ^ hates(z, y) → ¬date(z, x)

(b). CNF

1. own(You)
2. buys_carrot(Robin)
3. looking_for(You, Robin)
4. ¬ owns(x1, x2) v ¬ rabbit(x2) v ¬ rabbit(x3) v ¬ chases(x4, x3) v hates(x1, x4)
5. ¬ dog(b) v rabbit(F(b))
6. ¬ dog(b) v chases(b, F(b))
7. ¬ buys-carrot(z) v owns(z, G(z))
8. ¬ buys-carrot(z) v rabbit(G(z)) v grocery(G(z))
9. ¬ owns(a1, a2) v ¬ hates(a3, a2) v ¬ date(a3, a1)
10. dog(D)

(c) Translating the conclusion

FOL

¬ (∃x grocery(x) ^ own(Robin, x)) → ¬ date(Robin, You)

CNF

¬ (∃x grocery(x) ^ own(Robin, x)) ^ ¬ ¬ date(Robin, You) as ¬ (P → Q)

= P ^ ¬ Q

{∀x ¬ grocery(x) v ∀x ¬ owns(ROBIN, x)} ^ date(Robin, x)

{¬ grocery(y) v owns(Robin, y)} ^ date(Robin, You)

So,

1. ¬ grocery(y) v ¬ owns(Robin, y)
2. date(Robin, You)

(d). Proving the statement

{date(Robin, You)} ^ {¬ own(a1, a2) v ¬ hate(a3, a2) v ¬ date(a3, a1)} → {¬

own(You, a2) v ¬ hate(Robin, a2)}

{Robin/a3} and {You/a1}

{¬ own(You, a2) v ¬ hate(Robin, a2)} ^ {¬ own(x1, x2) v ¬ rabbit(x2) v ¬ rabbit(x3) ¬ chases(x4, x3) v hate(x1, x4)} → {own(You, a2) v own(Robin, x2) v rabbit(x2) v ¬ rabbit(x3) v ¬ chase(a2, x3)}

{Robin/x1} and {a2/x4}

{¬ own(You, a2) v ¬ hate(Robin, x2) v ¬ rabbit(x2) v ¬ rabbit(x3) v ¬ chase(a2, x3)} ^{own(You, D)} → {¬ own(Robin, x2) v ¬ rabbit(x2) v ¬ rabbit(x3) v ¬ chases(D, x3)}

{D/a2}

{¬ own(, x2) v ¬ rabbit(x2) v ¬ rabbit(x3) v ¬ chase(D, x3)} ^{¬ dog(b) v chases(b, F(b))} → {¬ own(Robin, x2) v ¬ rabbit(x2) v ¬ rabbit(F(D)) v ¬ Dog(D)}

{D/b} and {F(D)/x3}

{¬ own(R, x2) v ¬ rabbit(x2) v ¬ rabbit(F(D)) v ¬ dog(D)} ^{dog(b) v rabbit(F(b))} → {¬ own(Robin, x2) v ¬ rabbit(x2) v dog(D)}

{D/b}

{¬ own(Robin, x2) v ¬ rabbit(x2) v ¬ dog(D)} ^{dog(D)} −→ {¬ own(Robin, x2) v ¬ rabbit(x2)}

{¬ own(Robin, x2) v ¬ rabbit(x2)} ^{¬ buy_carrot(z) v rabbit(G(z)) v grocery(G(z))} → {¬ own(Robin, G(z)) v ¬ buy_carrot(z) v grocery(G(z))}

{G(z)/x2}

{¬ own(Robin, G(z)) v ¬ buy-carrot(z) v grocery(G(z))} ^{¬ grocery(y) v ¬ own(Robin, y)} −→ {¬ buy_carrot(z) v ¬ own(Robin, G(z))}

{G(z)/y}

{¬ buy_carrot(z) v ¬ own(Robin, G(z))} ^{buy_carrot(z) v owns(z, G(z))} → {¬ buy_carrot(ROBIN)}

{Robin/z}

{¬ buy_carrot(Robin)} ^{¬ buy_carrot(z)} → {¬looking_for(You, Robin)}

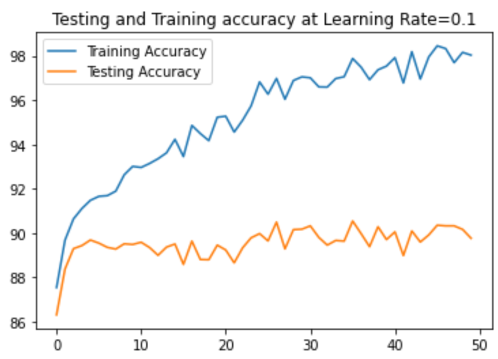
2. Lost in the Closet

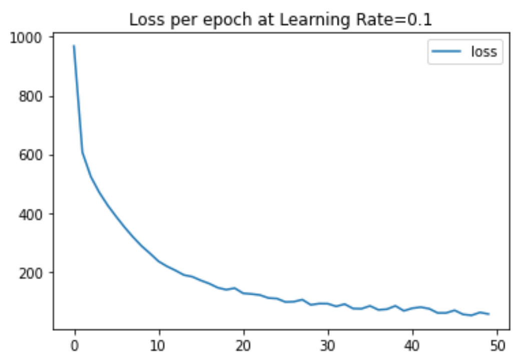
(a). Loss Function

In the given problem, one is required to differentiate between various fashion products which would require multi-class classification to differentiate them. The loss function used is Cross-Entropy which measures the difference between two probability distributions (actual and predicted). It can be defined as[1]:

$$\text{Cross-Entropy} = L(\mathbf{y}, \mathbf{t}) = -\sum_i \mathbf{t}_i \, \ln \mathbf{y}_i$$

(b). CNN






```
Loss: 55.3006, Train Accuracy: 98.04%, Test Accuracy: 89.76%
```

In the above figures, using a learning rate of 0.1, it gives us result with high training accuracy and testing accuracy and a low loss. There seems to be slight overfitting of data which indicates bad generalisation.

(c). Various Activation functions and different learning rates

Table 1 - Activation functions comparison

Activation function	Training Accuracy (%)	Testing Accuracy (%)	Loss
Tanh	97.19	89.60	54.7156
Sigmoid	98.00	90.22	12.3431
ELU	98.17	90.48	69.4989

In table 1, the ELU has given the best training and testing accuracies, however it has more loss. Sigmoid has the least loss.

Table 2 - ReLu different Learning Rates Comparison

Learning Rate	Training Accuracy (%)	Testing Accuracy (%)	Loss
0.001	88.80	87.48	607.8469
0.1	98.04	89.76	55.3006
0.5	89.11	85.14	625.6817
1	10.00	10.00	4330.5616
10	10.00	10.00	4636.2253

In table 2, a learning rate of 0.001 provides subpar results while taking more computing time. A learning rate of 0.1 provides the best results with the minimal loss. At learning rate of 0.5, the result is almost equivalent to the result of learning rate 0.001 but with faster computation but from this point onward the results seem to be becoming suboptimal. At learning rates 1 and 10, the training and testing accuracies both become 10% and it seems that there is no point in increasing learning rates as large learning rates give suboptimal results. Learning rates < 0.5 can be further explored to see if other values (i.e., 0.01, 0.2 etc) could provide a better than learning rate 0.1

(c). Different drop off

In the below figures, it shows the results with a drop rate of 0.3 to the second fully connected layer of the neural network with a learning rate of 0.1 The training accuracy has slightly decreased but the testing accuracy and the loss has almost doubled. There seems to be slight reduction in overfitting but at the cost of the loss doubling and the training accuracy decreasing.

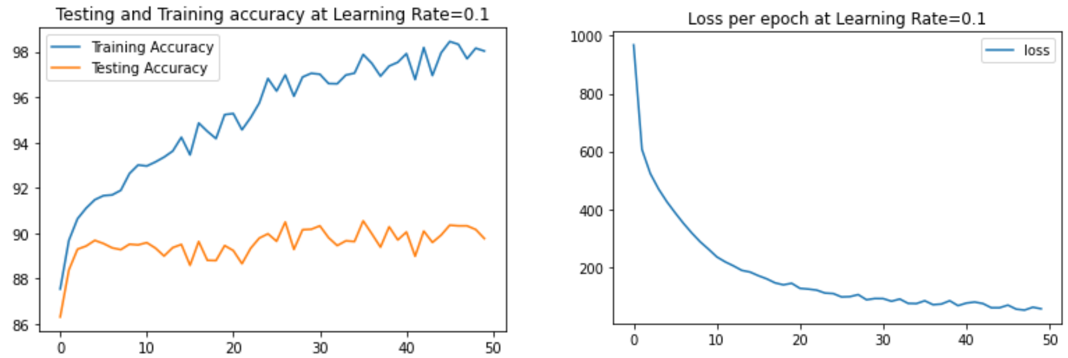


Figure 1 - Testing/Training Accuracy and Loss with dropout

In table 3, dropout of 0.1 gives the best result with the least loss, whereas higher dropout rates give lower training and testing accuracies, whilst doubling the loss.

Table 3 – Dropoff Comparison

Dropout	Training Accuracy (%)	Testing Accuracy (%)	Loss
0.1	98.04	90.58	74.9527
0.3	97.93	90.50	124.0854
0.6	97.40	91.30	215.1526

References

1. 365 Data Science. 2021. *What Is Cross-Entropy Loss? | 365 Data Science*. [online] Available at: <https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/>