CS7015 Deep Learning
Programming Assignment 5
RBMs

CS17S008 Nitesh Methani
CS17S013 Pritha Ganguly

April 30, 2018

Contents

List of Figures

1 Introduction

A restricted Boltzmann machine (RBM) is a generative stochastic artificial neural network that can learn a probability distribution over its set of inputs. RBMs are used for dimensionality reduction, classification, to name a few. They form a variant of Boltzmann machines, with the restriction that their neurons must form a bipartite graph: a pair of nodes from each of the two groups of units (commonly referred to as the "visible" and "hidden" units respectively) may have a symmetric connection between them; and there are no connections between nodes within a group.

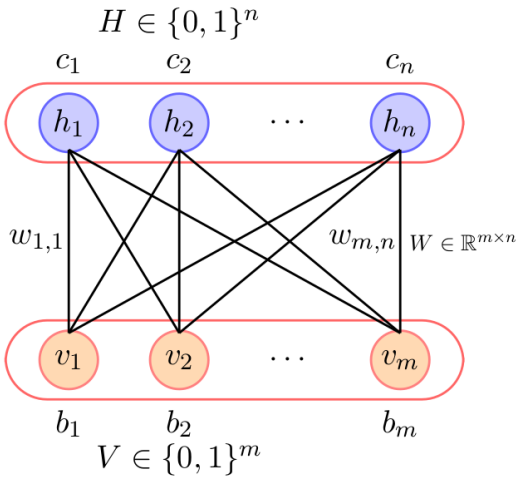


Figure 1: RBM with m visible units and n hidden units

The basic, single-step contrastive divergence procedure for a single sample can be summarized as follows:

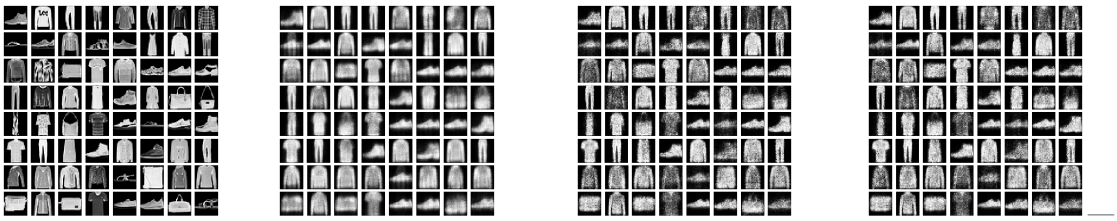
1. Take a training sample v, compute the probabilities of the hidden units and sample a hidden activation vector h from this probability distribution.
2. Compute the outer product of v and h and call this the positive gradient.
3. From h, sample a reconstruction v' of the visible units, then resample the hidden activations h' from this. (Gibbs sampling step)
4. Compute the outer product of v' and h' and call this the negative gradient.
5. Let the update to the weight matrix W be the positive gradient minus the negative gradient, times some learning rate:
$$\nabla W = \eta(vh^T - v'h'^T)$$
6. Update the biases a and b analogously:
$$\nabla a = \eta(v - v')$$
$$\nabla b = \eta(h - h')$$

2 Roadmap

In Section 3 we tried to visualize the reconstructed images for different dimensions of the latent space. In Section 4 we plotted the t-SNE representations of the hidden and visible units with different hyperparameter settings and also compared the execution time. We conclude this assignment in Section 6 after listing some observations in Section 5.
As it is our last assignment, in Section 7 we have also plotted some results form the other experiments that we encountered during the course.

3 Visualization

If we choose a low-dimensional latent space, we can use the learned representations to project high-dimensional data to a low-dimensional manifold. We experimented with different dimensions for the latent space and the resulting images generated are plotted below



(a) Input Image (b) 50-D Latent Space (c) 300-D Latent Space (d) 500-D Latent Space

Figure 2: Random samples from learned generative models (RBM) of Fashion MNIST for different dimensionalities of latent space.

We can see from the Figure 2 that as we increase the dimensions of the latent space, the images generated are more accurate.

4 Experiments

For the experiments on image generation, we have used the Fashion MNIST dataset. The labels of different classes are as follows :

0	1	2	3	4	5	6	7	8	9
T-shirt/Top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot

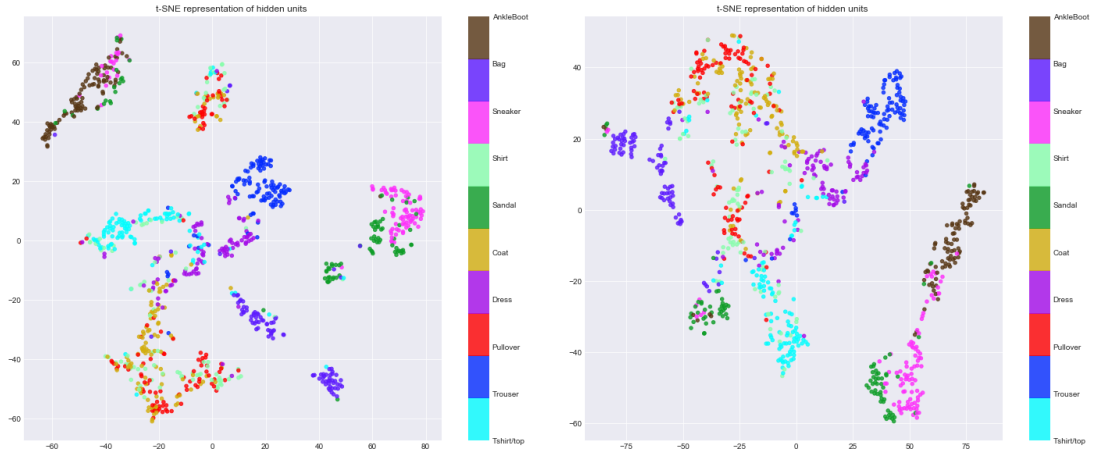
These class indices have been used on the colour-bars in the subsequent plots.

4.1 Effect of different values of h

Hidden Units

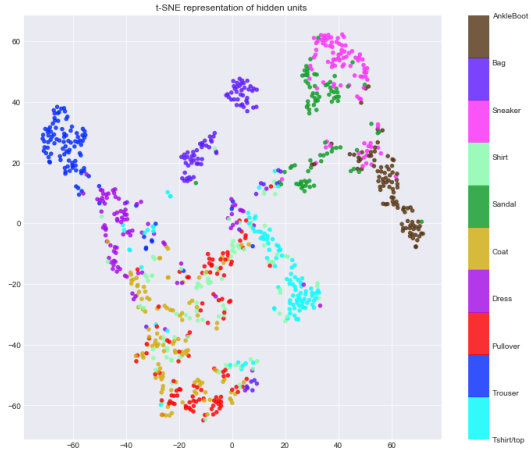
From the following t-SNE representations we can deduce the following observations:

- The clusters of classes Coat, Pullover and Shirt (Class 4, Class 2, Class 6 respectively) have the most overlap. See Figure 2(a).
- Similarly, classes Ankleboot, Sneaker and Sandal (Class 9, Class 5, Class 7 respectively) are clustered together. See Figure 2(a).
  It is because these classes share the similar latent representation.
- As the latent space dimension is increased, the classes are getting well separated and the overlap is less. See Figure 2(d).
- With the increase in value of h, we can observe the significant decrease in the training loss. See Figure 2(e).
- Also note that as we increase the value of h the training time is also increasing. See Figure 2(f).

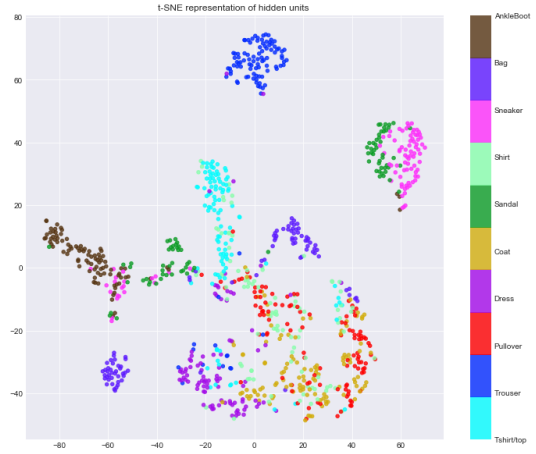


(a) h = 50

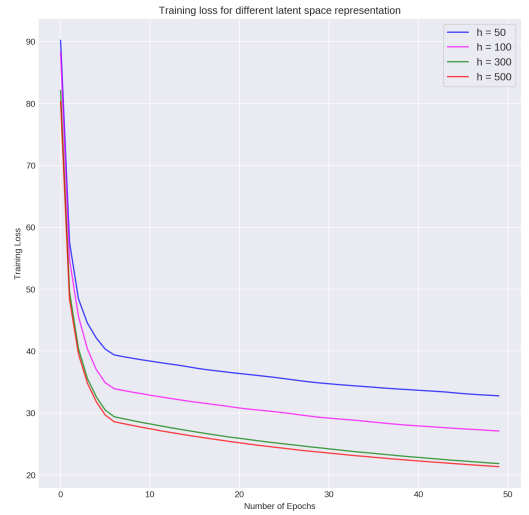
(b) h = 100



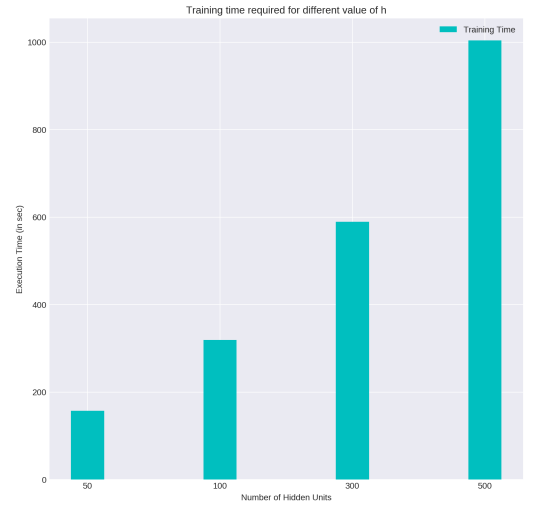
(c) h = 300



(d) h = 500



(e) Training Loss



(f) Training Time

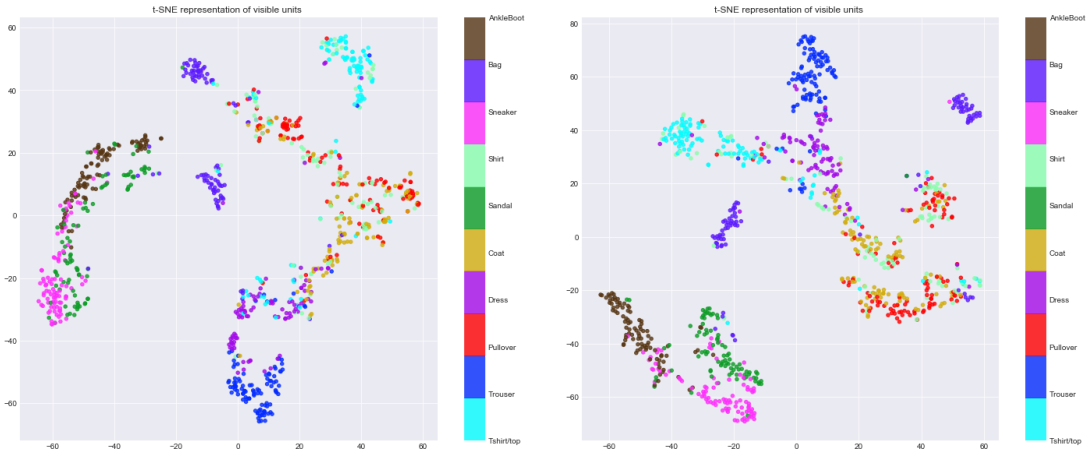
Figure 3: Experiments showing the effect of different latent space dimensions

Visible Units

From the following t-SNE representations we can deduce the following observations.

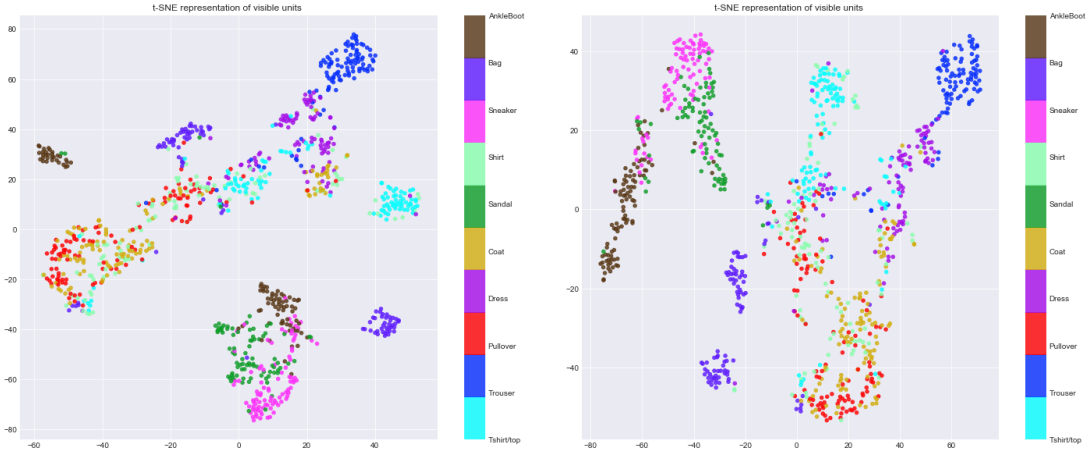
- The clusters of classes Coat, Pullover and Shirt (Class 4, Class 2, Class 6 respectively) have the most overlap. See Figure 5(a).

- Similarly, classes Sneaker and Sandal (Class 7, Class 5 respectively) are clustered together. See Figure 4(a).
  It is because for these classes our model has learned similar abstract features.
- Classes Bag and Trouser (Class 8, Class 1 respectively) are very well separated.



(a) h = 50

(b) h = 100



(c) h = 300

(d) h = 500

Figure 4: Experiments showing the distribution of visible units reconstructed using different latent space dimensions

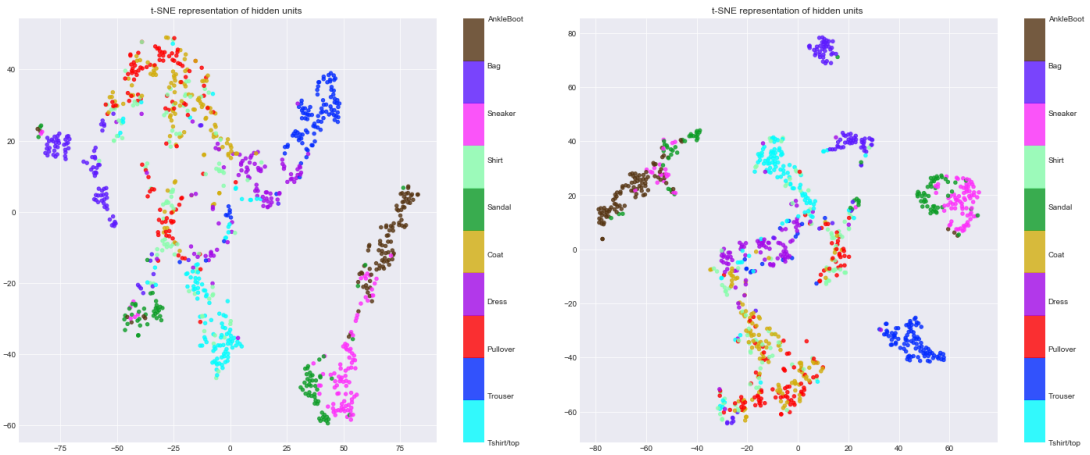
4.2 Effect of different values of k

Hidden Units

From the following t-SNE representations we can deduce the following observations. (Note that here $h = 100$).

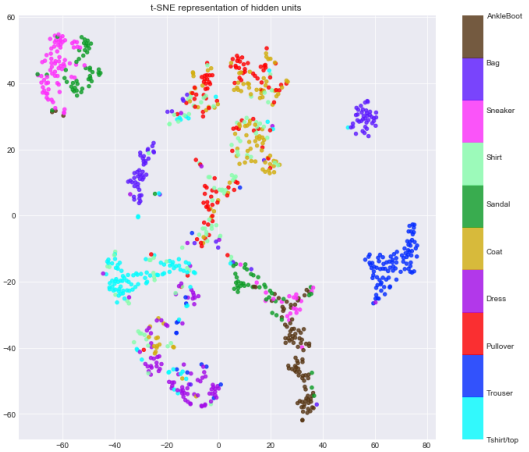
- The clusters of classes Coat, Pullover and Shirt (Class 4, Class 2, Class 6 respectively) have the most overlap. See Figure 4(a).
- Similarly, classes Ankleboot, Sneaker and Sandal (Class 9, Class 5, Class 7 respectively) are clustered together. See Figure 4(a).
  It is because these classes share the similar latent representation.
- Increasing the number of samples have very little effect on the overlapping clusters. See Figure 4(c).
- With the increase in value of k, we can observe that there is no significant improvement in the training loss. Keeping $k = 1$ works fine. See Figure 4(d).

- Also note that as we increase the value of k the training time is also increasing. See Figure 4(e).

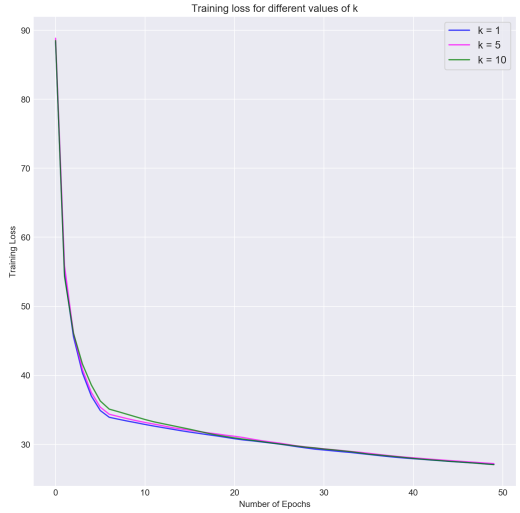


(a) k = 1

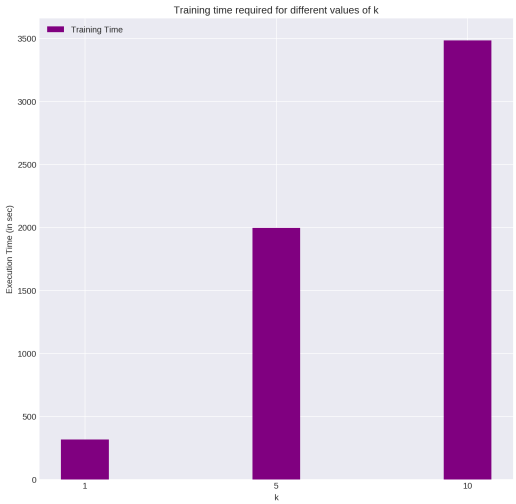
(b) k = 5



(c) k = 10



(d) Training Loss



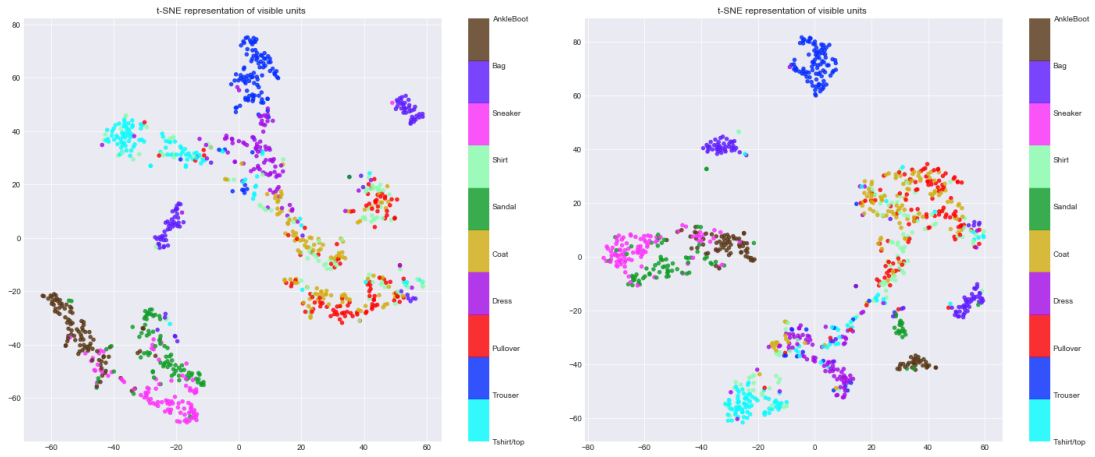
(e) Training Time

Figure 5: Experiment showing the effect of using different values of k in Gibbs Chain

Visible Units

From the following t-SNE representations we can deduce the following observations. (Note that here $h = 100$).

- The clusters of classes Coat, Pullover and Shirt (Class 4, Class 2, Class 6 respectively) have the most overlap. See Figure 5(a).
- Similarly, classes Sneaker and Sandal (Class 7, Class 5 respectively) are clustered together. See Figure 4(a).
  It is because for these classes our model has learned similar abstract features.
- Classes Bag, Trouser, Tshirt/Top and Ankleboot (Class 8, Class 1, Class 0, Class 9 respectively) are very well separated. (Note that Ankleboot class was overlapping with Sandal and Sneaker class in latent space.)
- Again here we can observe that there is not much difference in the reconstructed images with the increased value of k.



(a) k = 1

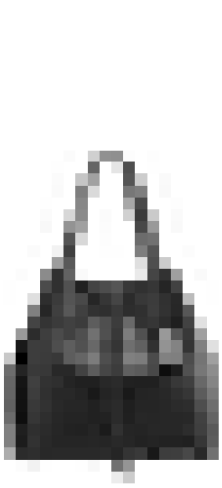
(b) k = 5

(c) k = 10

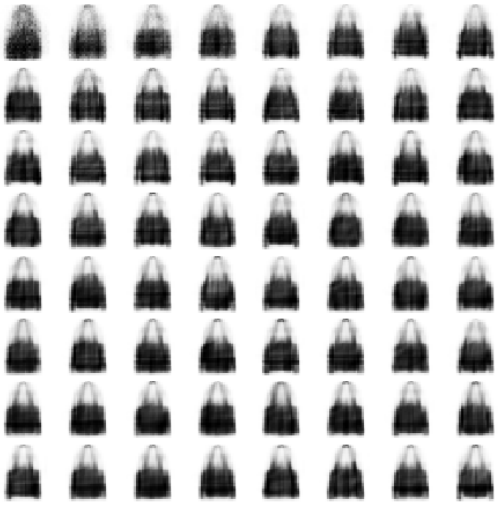
Figure 6: Experiment showing the effect of using different values of k in Gibbs Chain on the reconstructed images

4.3 Samples generated by Gibbs Sampling

Contrastive Divergence algorithm took $m = 3840$ iterations of SGD to converge. We have plotted the samples $\frac{m}{64}$ steps of SGD. The 8×8 below shows the samples generated by Gibbs chain after every 64 steps.

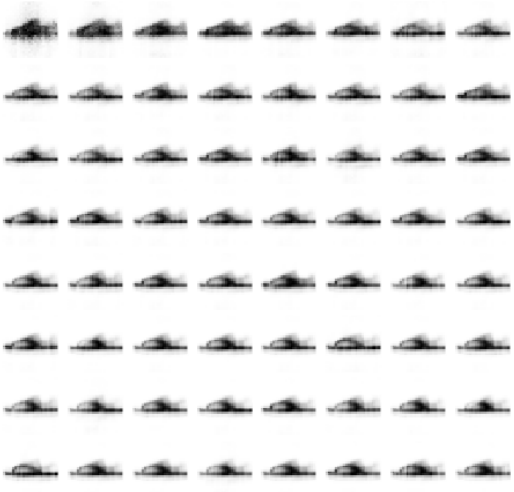


(a) Original



(b) Samples generated

(c) Original



(d) Samples generated

Figure 7: Samples generated by Gibbs chain after every $\frac{m}{64}^{th}$ step where $m = 3840$

We can see that in initial iterations, the samples were a bit noisy but as the training progresses we get better and better samples.

5 Conclusion

In this assignment we implemented RBMs from scratch without using any Neural Net frameworks. We experimented with different values of latent-space dimensions, h, and the number of samples generated, k, by Gibbs chain. We observed that the reconstruction error is decreasing gradually as the value of h increases. This is because network gets more degree of freedom to model the input data. We also observed that $k = 1$ works fine in practice. During the course of this assignment we were exposed to different techniques like Gibbs Sampling, Contrastive divergence, etc. We gained practical experience with the training and managed to produce reasonably good results.

6 Cool Stuffs

6.1 YOLO

In this experiment we tried to detect objects with the YOLO system using a pre-trained model. Few examples are show below.

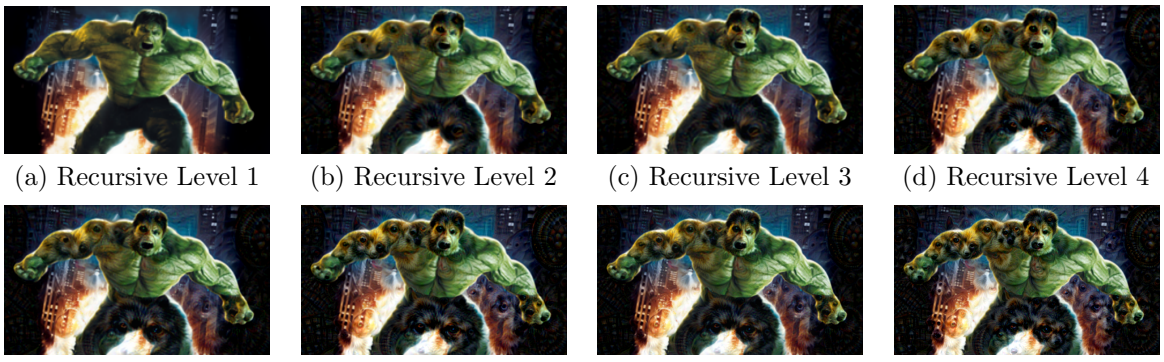
Figure 8: Object detection using YOLO

As we can see some fine-tuning is required. But the results are pretty decent.

6.2 Deep Dream

In this experiment we used the gradient of the neural network to amplify patterns in the input image. This is commonly called the DeepDream algorithm. We use the Inception model for it. This experiment helped us to understand how to use the gradient of a neural network to amplify patterns in an image. The output images appeared to have been redrawn with abstract or animal-like patterns. Few examples are shown below. Note how the colours of the original image are mostly kept in the DeepDream images. This is because the gradient is blurred in its colour-channels so it becomes somewhat gray-scale and mainly changes the shape of the image and not so much its colour.

6.2.1 The Incredible Hulk

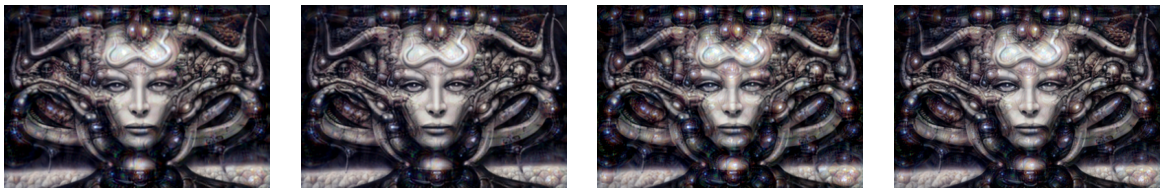


(a) Recursive Level 1 (b) Recursive Level 2 (c) Recursive Level 3 (d) Recursive Level 4

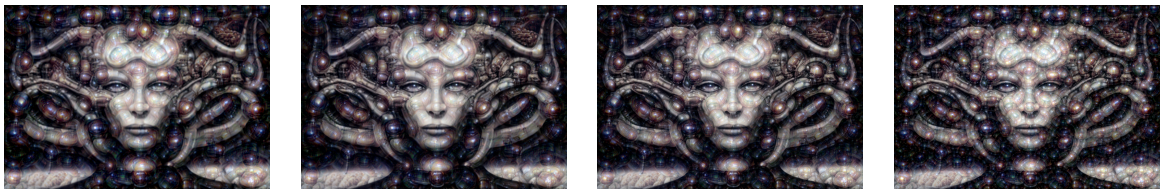
(e) Recursive Level 5 (f) Recursive Level 6 (g) Recursive Level 7 (h) Recursive Level 8

Figure 9: DeepDreaming on The Incredible Hulk

6.2.2 Giger



(a) Recursive Level 1 (b) Recursive Level 2 (c) Recursive Level 3 (d) Recursive Level 4



(e) Recursive Level 5 (f) Recursive Level 6 (g) Recursive Level 7 (h) Recursive Level 8

Figure 10: DeepDreaming on the witch Giger

6.3 Image Captioning

In this experiment we tried to understand how to generate captions using a pre-trained image-model (VGG16). It generates a "thought-vector" of what the image contains, and then we trained a Recurrent Neural Network to map this "thought-vector" to a sequence of words.

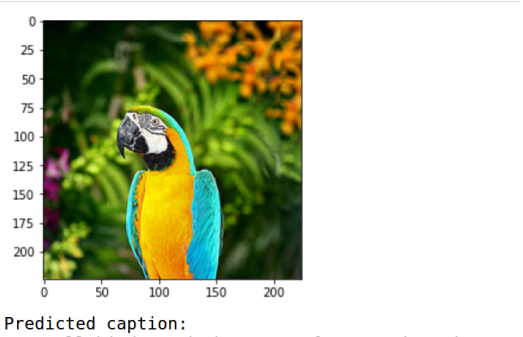


Figure 11: Image Captioning

References

[1] Mitesh Sir's slides, *course slides*

[2] RBM Wikipedia
https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine#Training_algorithm

[3] RBM tutorial
http://deeplearning.net/tutorial/rbm.html

[4] Havss Lab
https://github.com/Hvass-Labs/TensorFlow-Tutorials