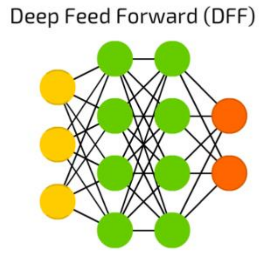
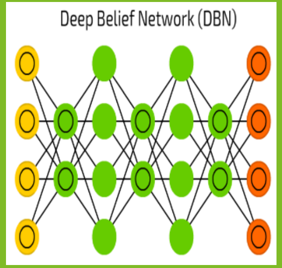
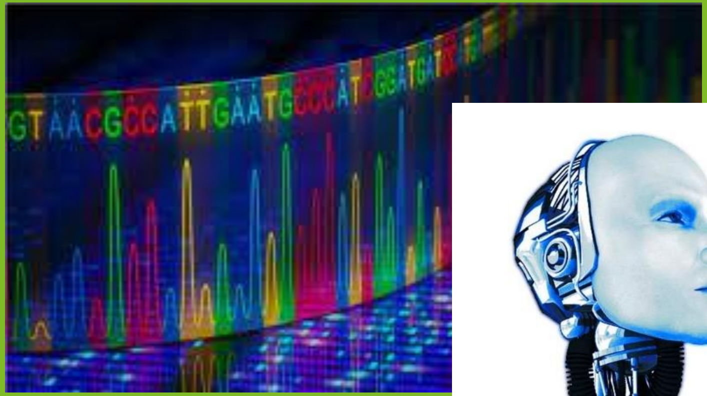
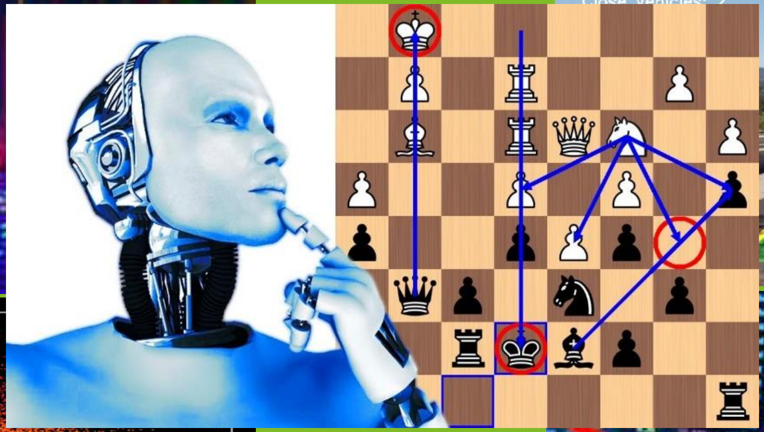
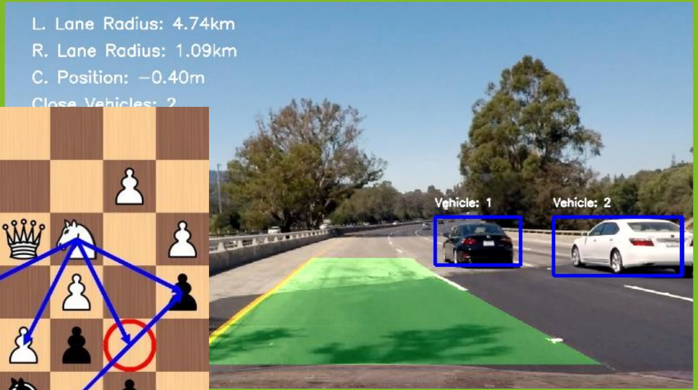
Deep Learning CSI_7_DEL

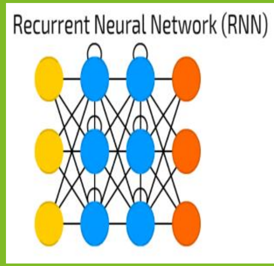


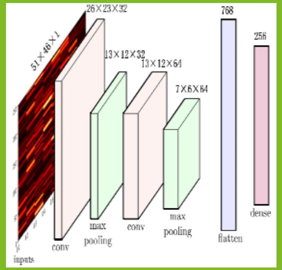


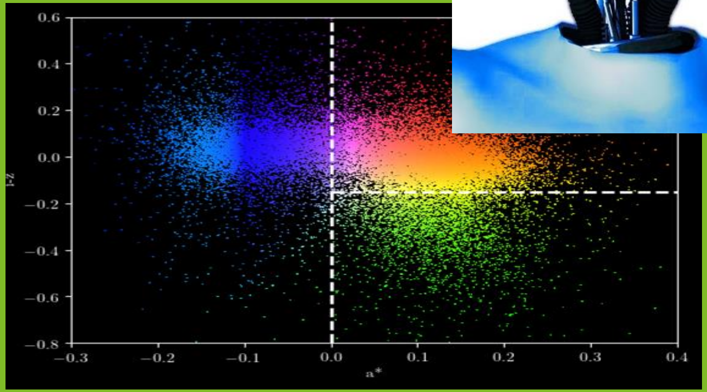


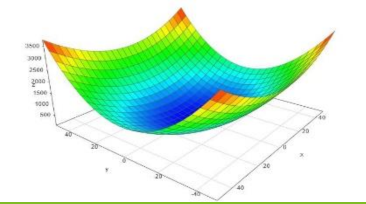












Week 9:Deep Belief Networks

London South Bank University
EST 1892

Deep Belief Networks

- Probabilistic generative model.
- Introduced by Hinton et al in 2006.
- Non-linear dimensionality reduction.
- Captures correlations between the activities of the hidden features in the preceding layers.
- A Greedy layer-by-layer unsupervised training (Vanishing gradient?)

London South Bank University
EST 1892

Deep Belief Networks: Characteristics

- **Generative:** DBN can produce randomly created values for the input values. Some research paper refer to this as **dreaming.**
- **Probabilistic:** DBNs are used for classification tasks. The output is the probability that certain input belongs a particular class.
- **Multi-layered:** like other neural networks, DBN is made up of multiple layers.
- **Stochastic latent variables:** Since DBN is made up of stacked RBMs, it produces random (*stochastic*) values that can not be directly observed (*latent*).

London South Bank University
EST 1892

DBN vs Feedforward Neural Networks

Deep Belief Networks	Feedforward Neural Networks
Input must be **Binary**	Input can be **decimal** or **binary**
The output is **a class** to which the input belongs	The output can be **a class** or a **numeric** prediction
Can generate **plausible input** based on a given outcome	Can not perform like DBN

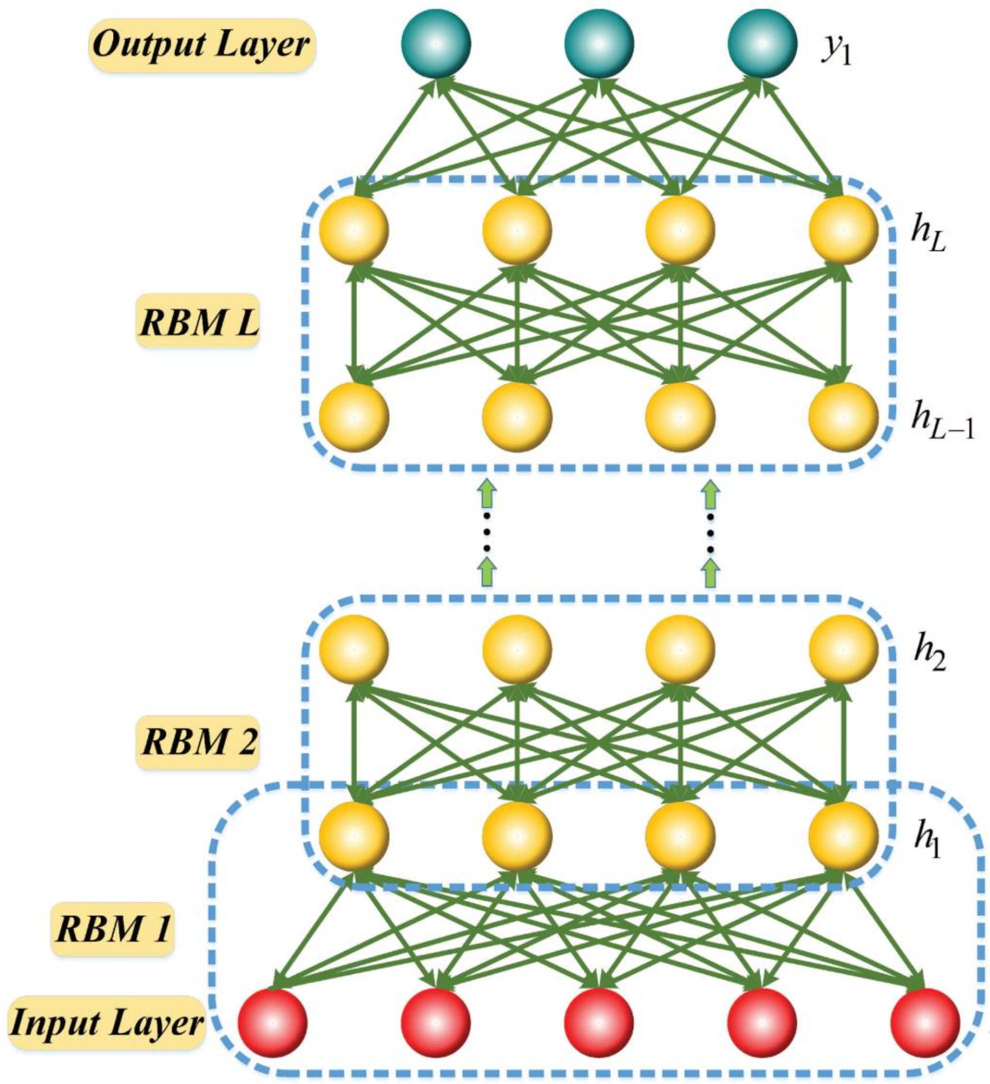
DBN Applications

- Generating and reconstructing images (Hinton et al 2007)
- Collaborative filtering for recommender system(Salakhutdinov et al., 2007).
- Motion-capture data (Taylor et. al. 2007).
- Images and information retrieval and reconstruction. (Gehler et al. 2006)

London South Bank University
EST 1892

DBN Architecture

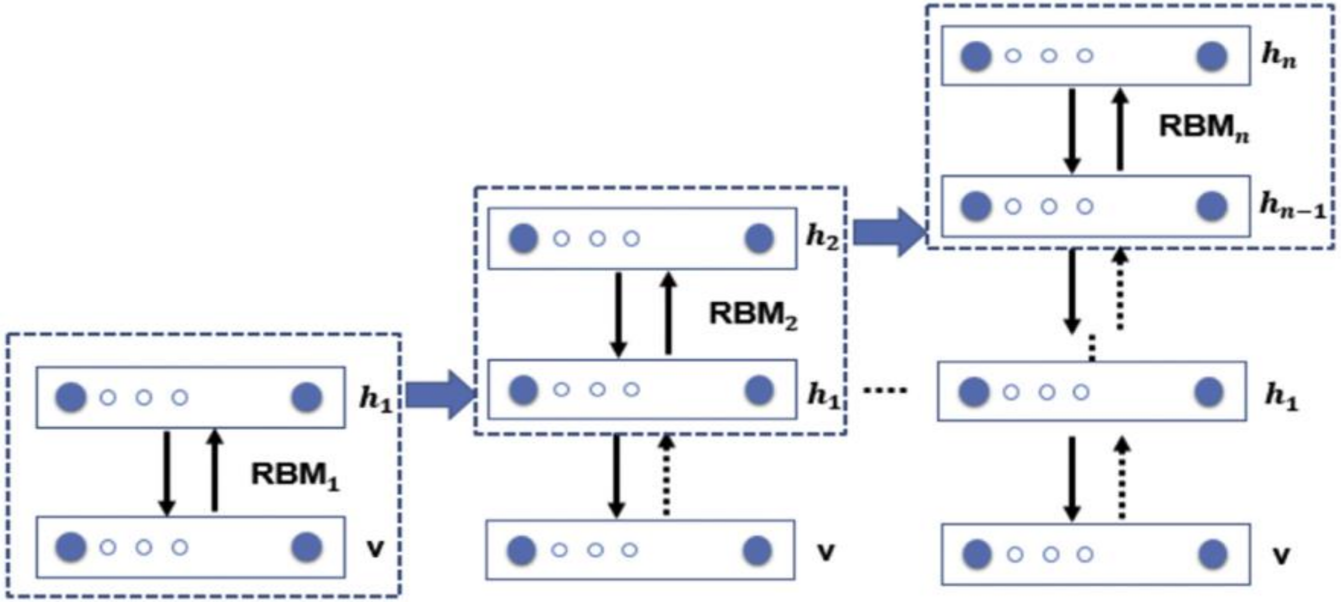
- Input fed to the DBN network passes through a series of layers.
- DBN is made up of **stacked RBM**
- **The hidden units** become the **output to the next layers**
- Adding additional RBMs causes deeper DBN.
- Even though RBMs are unsupervised, the desired **outcome in DBN is supervised**.
- A final logistic regression layer is included to associate the given input to an output class.



Source: https://www.mdpi.com/2072-4292/14/6/1484/htm

DBN Architecture: Stacked RBMs

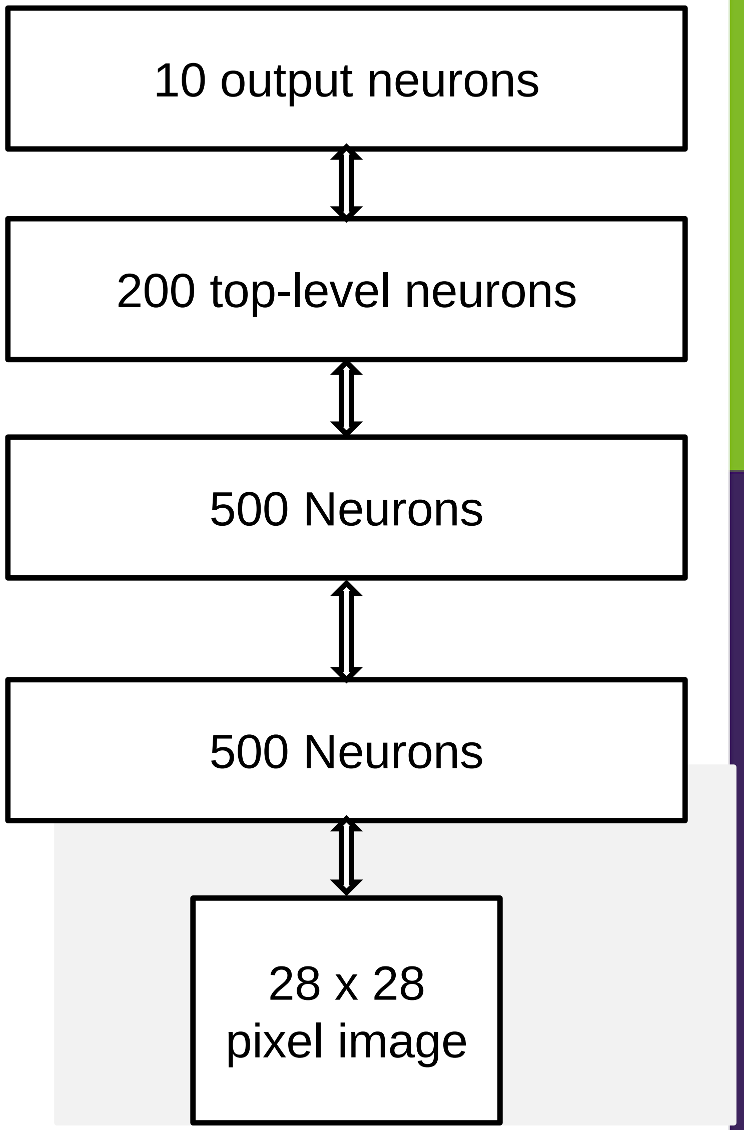
- Each RBM has visible layer v and a single layer h.
- RBM_1 is trained using the visible neurons
- The hidden layer h_2 of RBM_2 is trained using the previously trained layer h_1.
- The output of $h_2$is used to train RBM_3 and son on.



Source: https://www.sciencedirect.com/science/article/pii/B9780128154809000116

Early DBN: Hinton's 2006 Experiment

- First DBN network was trained by Hinton in 2006
- The input was a 28X28 pixels or 784 single bit vector
- Monochrome and single channel (black & white)
- 3 layers of stacked RBMs
- L1= 500 neurons, L2= 500 neurons, L3= 200 neurons
- The output neurons here are 10 digits.



DBN Dreaming of Digits

- The following digits have been created using DBN and were taking from Hinton’s (2006) deep learning paper
- The first row shows zeros generated by DBN using contrastive divergence by Gibbs sampling.

Source: Hinton et al 2006

London South Bank University
EST 1892

Energy function / Cost function

- **Low energy** translates to **high probability** (High accuracy).
- In the case of DBN, **there is an energy function for each RBM layer**.
- The total energy of the joint configuration of the visible and hidden neurons can be formulated:

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j$$
$$= -a^T v - b^T h - v^T W h$$

- Where a_i, b_i are biases, weights represents w_{ji} and v_i, b_j are the corresponding visible and hidden units.

London South Bank University

EST 1892

Recap: RBM: Hidden and Visible units

- In For any hidden unit h_i, he probability P that h_i can be turn on can be computed as follow:

$$P(h_i = 1) = \frac{1}{1-e^{-z_i}}$$

(a sigmoid basically also called partition function)

z_i represents the sum of all input combinations:

$$\mathrm{z_i} = \sum_j w_{ji} v_i + b_i$$

Where w_{ji} is the weight connection between the visible and the hidden neurons and b_i is the bias added to the visible neuron.

- Similarly, for any hidden unit v_i, he probability P that v_i can be turn on can be computed as follow:

$$P(v_i = 1) = \frac{1}{1-e^{-y_i}}$$

$$Where\ y_i = \sum_j w_{ji} v_i + b_i$$

- Once this output is calculated, it gets passed on the next layer as input

London South Bank University

EST 1892

DBN Training

- DBN is pretrained using an algorithm called **Greedy layer-wise training**.
- **Each RBM layer is trained separately** with gradient descent.
- The purpose of this algorithm is to **produce binary vector to feed into the contrastive divergence** algorithm.
- The weights for each layer are updated using:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial \log p(v)}{\partial w_{ij}}$$

$$\frac{\partial \log p(v)}{\partial w_{ij}} = < v_i h_j >^0 - < v_i h_j >^\infty$$

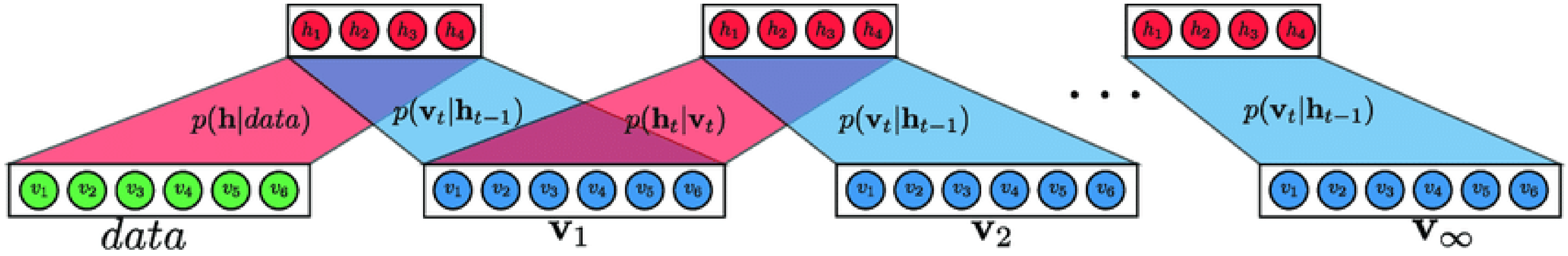
- Where $< v_i h_j >$ is the average over many generated samples

London South Bank University

EST 1892

DBN Training:

- Several steps of Gibbs sampling executed on each RBM layer.
- The logistic layer is trained using backpropagation.
- The whole network weights are adjusted in a single passed (fine-tuning)



London South Bank University
EST 1892

Contrastive Divergence by Gibbs sampling

- RBMs layers in DBN are pretrained using CD.
- The CD algorithm works as follow:

Step 1: $Initialise\ the\ visible\ units\ using\ a\ random\ training\ vector$

Step 2: $Start\ Gibb\ Sampling\ and\ repeat\ for\ N\ step$:

Step 2.1: $Update\ the\ hidden\ neurons\ given\ the\ visible\ neuron\ \ p\left(h_j = 1 \middle| V\right) = \sigma\left(b_j + \sum_i v_i\, w_{ij}\right)$

Step 2.2: $Update\ the\ visible\ neurons\ given\ the\ hidden\ neuron\ p(v_i = 1|H) = \sigma\left(a_i + \sum_j h_j w_{ij}\right)$

Step 2.3: $Re{-}update\ the\ hidden\ neurons\ given\ the\ constructed\ visible\ neurons\ using\ \ equation$

$\text{in } 2.1$

Step 2.4: $Update\ the\ weight W_{new} = W_{old} + \Delta W$

- Once this is done, DBN feeds the output of the current RBM hidden layer to the next RBM input layer as an input.

London South Bank University

EST 1892

Example: MNIST Dataset

Logistic layer — 10 output neurons

⇕

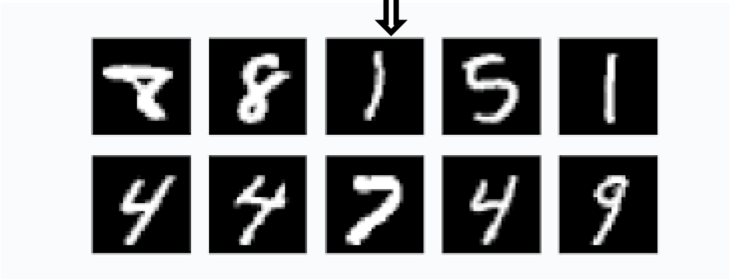
RBM 2 — 256 Neurons

⇕

RBM 1 — 256 Neurons

⇕

Input



Example: Greedy Algorithm Training

Layer-by-layer Greedy Training

```
[BernoulliRBM] Iteration 1, pseudo-likelihood = -26.72, time = 0.07s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -26.19, time = 0.10s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -24.22, time = 0.10s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -23.68, time = 0.11s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -22.78, time = 0.11s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -22.26, time = 0.12s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -22.44, time = 0.10s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -22.40, time = 0.11s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -21.67, time = 0.12s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -21.30, time = 0.13s
```

RBM 1 training

```
[BernoulliRBM] Iteration 1, pseudo-likelihood = -45.36, time = 0.11s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -44.76, time = 0.20s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -43.70, time = 0.17s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -44.89, time = 0.18s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -43.69, time = 0.18s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -42.48, time = 0.17s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -43.42, time = 0.17s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -43.12, time = 0.17s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -43.96, time = 0.16s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -44.07, time = 0.16s
```

RBM 2 training

Example: Model performance

```
Model performance:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        27
           1       0.93      0.74      0.83        35
           2       0.79      0.86      0.83        36
           3       0.86      0.83      0.84        29
           4       0.97      0.97      0.97        30
           5       0.97      0.97      0.97        40
           6       1.00      1.00      1.00        44
           7       0.93      0.97      0.95        39
           8       0.85      0.85      0.85        39
           9       0.82      0.88      0.85        41

    accuracy                           0.91       360
   macro avg       0.91      0.91      0.91       360
weighted avg       0.91      0.91      0.91       360
```

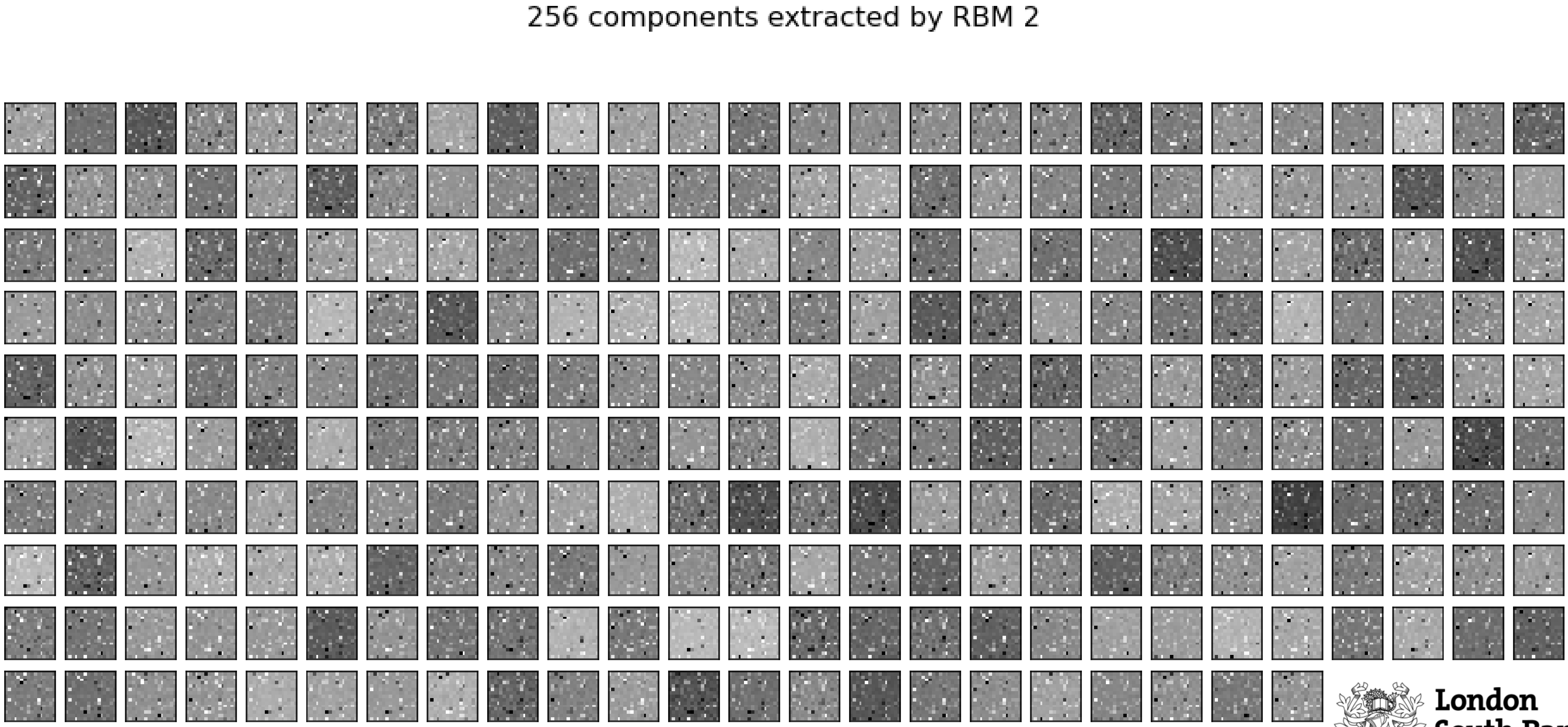
London South Bank University
EST 1892

Example: RBM 1 Components

256 components extracted by RBM 1

London South Bank University
EST 1892

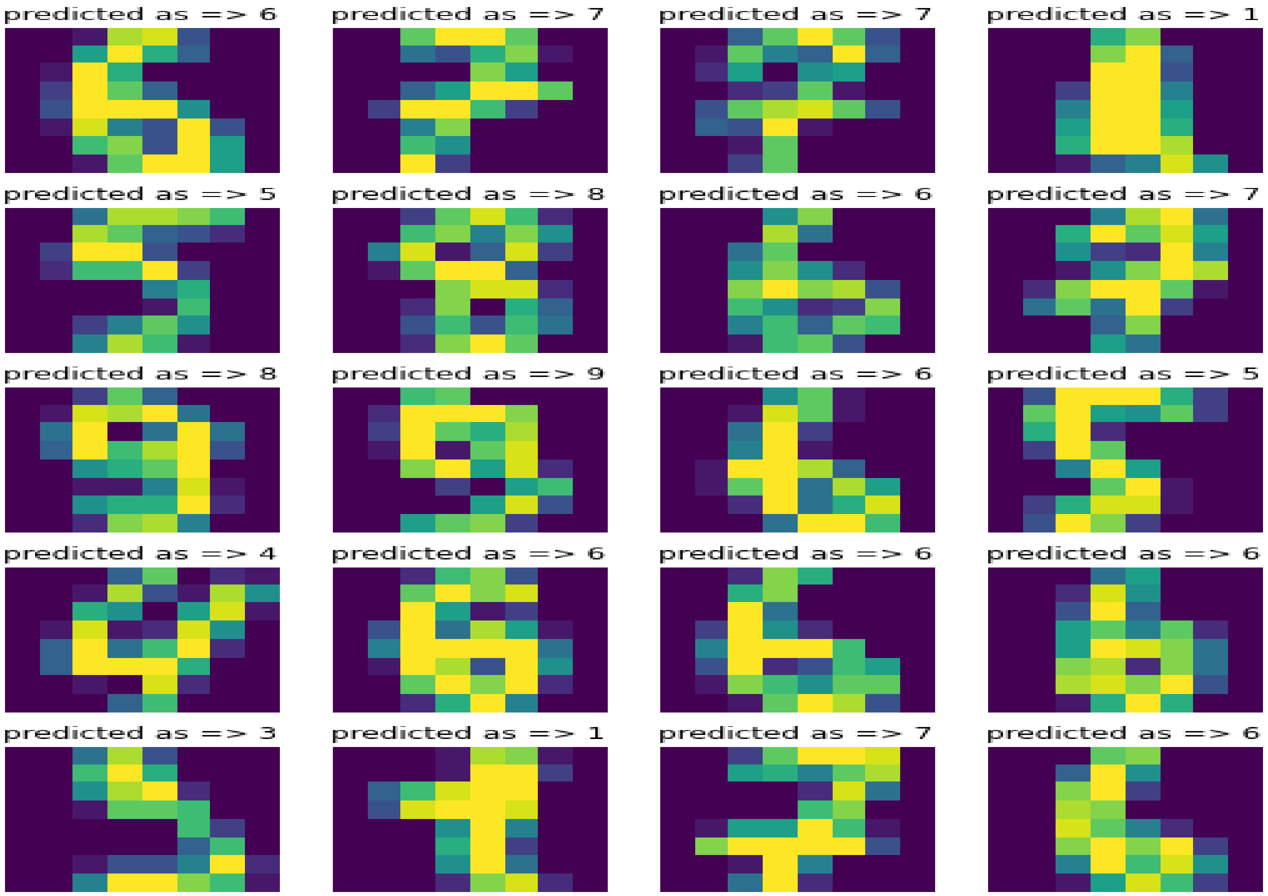
Example: RBM 2 components



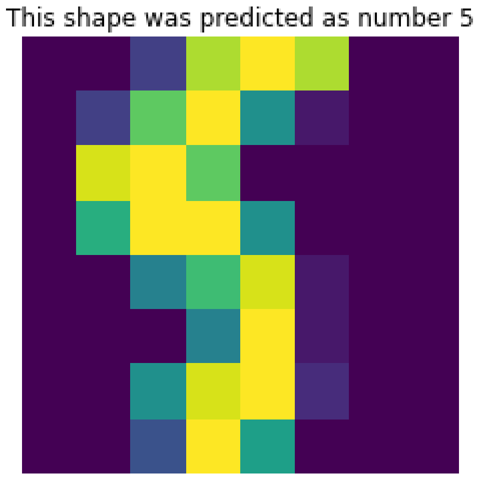
London South Bank University
EST 1892

Example: Logistic Classification

Multi input classification results



Single classification result



London South Bank University
EST 1892

DBN Limitations

- Very restrictive!
- Sometimes **slow to train** given the number of samples.
- The input **can only be binary** and not continuous.
- DBNs **can only be used for classification** and not for regression

London South Bank University
EST 1892

Summary

- A DBN network undergo unsupervised and supervised training.
- During the unsupervised training, DBN does not use the output labels.
- During the supervised training, on training data with labels is used.
- Once the unsupervised phase is finished, the output from the layers is refined with supervised logistic regression.
- The logistic function layer is used for classification task.

London
South Bank
University
EST 1892

Questions & Answers

London South Bank University
EST 1892