- Recurrent networks
  - Dynamic system uses the idea behind bigram models, and uses the same transition function over time:
    ⇨ $a_{t+1} = f_a\left(a_t, x_{t+1}\right)$ and $y_{t+1} = f_o\left(a_{t+1}\right)$
    ⇨ $a_{t+2} = f_a\left(a_{t+1}, x_{t+2}\right)$ and $y_{t+2} = f_o\left(a_{t+2}\right)$
    ⇨ $a_{t+3} = f_a\left(a_{t+2}, x_{t+3}\right)$ and $y_{t+3} = f_o\left(a_{t+3}\right)$
    ⇨ ...
  - Given input $x_{i,t,j}$ for item $i = 1, 2, \ldots, n$, time $t = 1, 2, \ldots, t_i$, and feature $j = 1, 2, \ldots, m$, the activations can be written as $a_{t+1} = g\left(w^{(a)} \cdot a_t + w^{(x)} \cdot x_t + b^{(a)}\right)$.
    ⇨ Each item can be a sequence with different number of elements t_i, therefore, each item has different number of activation units $a_{i,t}$, $t = 1, 2, \ldots, t_i$.
    ⇨ There can be either one output unit at the end of each item $o = g\left(w^{(o)} \cdot a_{t_i} + b^{(o)}\right)$, or t_i output units one for each activation unit $o_t = g\left(w^{(o)} \cdot a_t + b^{(o)}\right)$.
  - Multiple recurrent layers can be added where the previous layer activation $a_t^{(l-1)}$ can be used in place of x_t as the input of the next layer $a_t^{(l)}$, meaning $a_{t+1}^{(l)} = g\left(w^{(l)} \cdot a_t^{(l)} + w^{(l-1)} \cdot a_{t+1}^{(l-1)} + b^{(l)}\right)$.
  - Neural networks containing recurrent units are called recurrent neural networks: Wikipedia.
    ⇨ Convolutional layers share weights over different regions of an image.
  - ⇨ Recurrent layers share weights over different times (positions in a sequence).
- Bp through time
  - The gradient descent algorithm for recurrent networks are called Backpropagation Through Time (BPTT): Wikipedia.
  - It computes the gradient by unfolding a recurrent neural network in time.
    ⇨ In the case with one output unit at the end, $\dfrac{\partial C_i}{\partial w^{(o)}} = \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial w^{(o)}}$, and
    $\dfrac{\partial C_i}{\partial w^{(a)}} = \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial w^{(a)}} + \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial a_{t_i-1}}\dfrac{\partial a_{t_i-1}}{\partial w^{(a)}} + \ldots + \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial a_{t_i-1}} \cdots \dfrac{\partial a_2}{\partial a_1}\dfrac{\partial a_1}{\partial w^{(a)}}$, and
    $\dfrac{\partial C_i}{\partial w^{(x)}} = \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial w^{(x)}} + \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial a_{t_i-1}}\dfrac{\partial a_{t_i-1}}{\partial w^{(x)}} + \ldots + \dfrac{\partial C_i}{\partial o_i}\dfrac{\partial o_i}{\partial a_{t_i}}\dfrac{\partial a_{t_i}}{\partial a_{t_i-1}} \cdots \dfrac{\partial a_2}{\partial a_1}\dfrac{\partial a_1}{\partial w^{(x)}}$.
    ⇨ The case with one output unit for each activation unit is similar.
  - Repeat for each output if the output is a sequence
- Vanishing and exploding gradient
  - Vanishing gradient: if the weights are small, the gradient through many layers will shrink exponentially to 0
  - Exploding gradient: if the weights are large, the gradient through many layers will grow exponentially to ±∞.
  - In recurrent neural networks, if the sequences are long, the gradients can easily vanish or explode.
  - In deep fully connected or convolutional networks, vanishing or exploding gradient is also a problem
- Gated recurrent unit and long short term memory
- Hopfield networks
  - Used to store memories in local mins of the network
  - Activation: given the weights and an initial set of activations (noisy input), the activation values can be updated to minimize the energy $E = -\dfrac{1}{2}\sum_{ij} w_{ij} a_i a_j - \sum_i b_i a_i$ to recover the memory: Link.
  - Energy function is the landscape that stores memory
  - Activation - a_i is randomly chosen and updated to 1 if $\sum_{j \neq i} w_{ij} a_j + b_i > 0$ and -1 otherwise
  - Training
    - Biases are not updated
- Generative adversarial network

    - Generator - input is noise, output is fake image
    - Discriminator -= input the fake or real image, output is binary class whether the images real
    - Same but opposite loss function - generator try to maximize the loss of the discriminator, the discriminator try to minimize the loss of classifying fake/real
- Diffusion
    - Train image to noise
    - Generate noise to image
- Transformers
    -