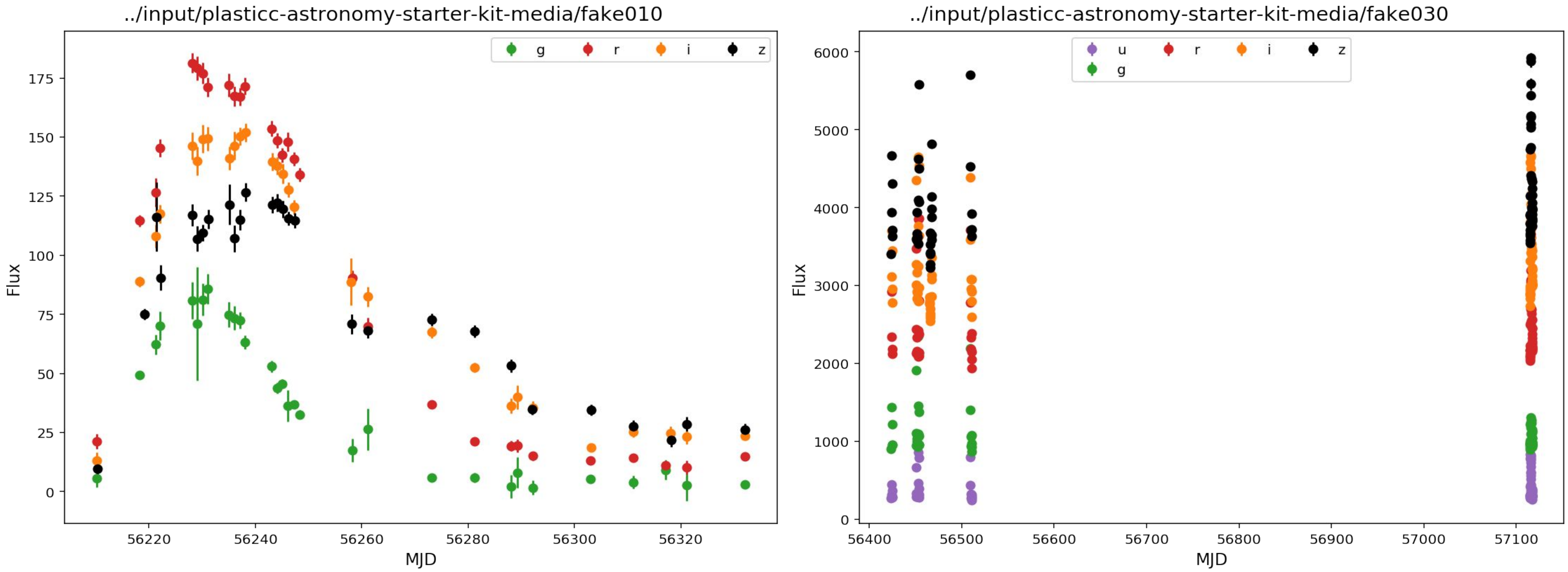
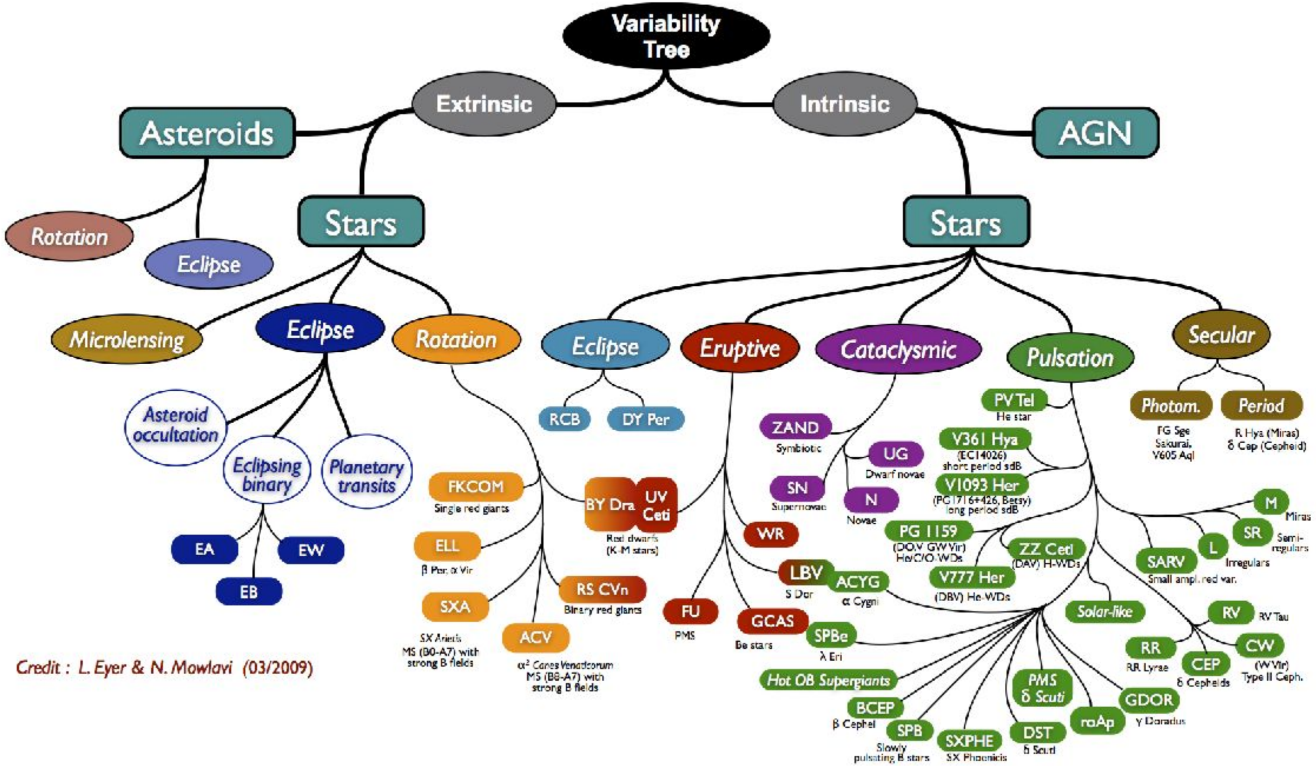
The time series analysis of the LSST

Carolina Cuesta-Lazaro
Machine Learning Journal Club

Flux : Brightness of the source as a function of time



Classify astronomical transient sources according to their flux



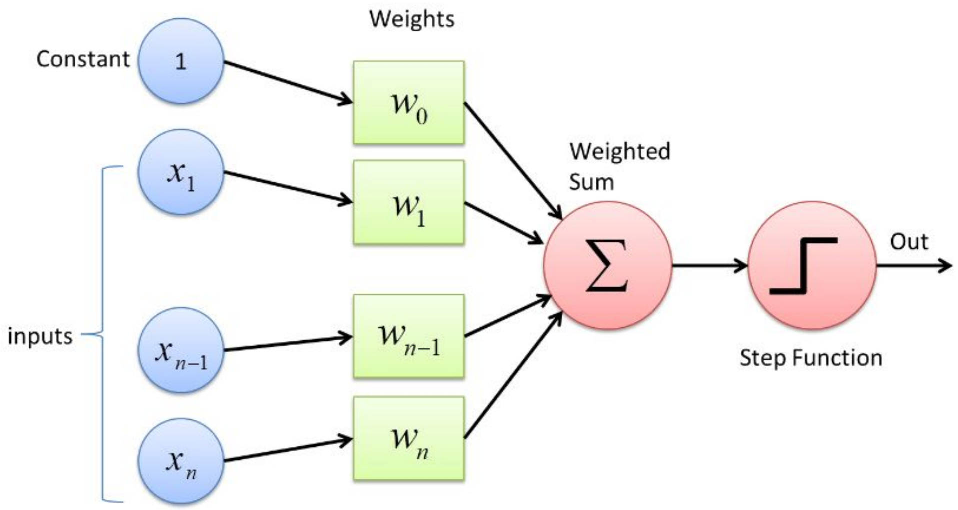
First attempt : Single Neuron

f(Inputs | Parameters)

Redshift

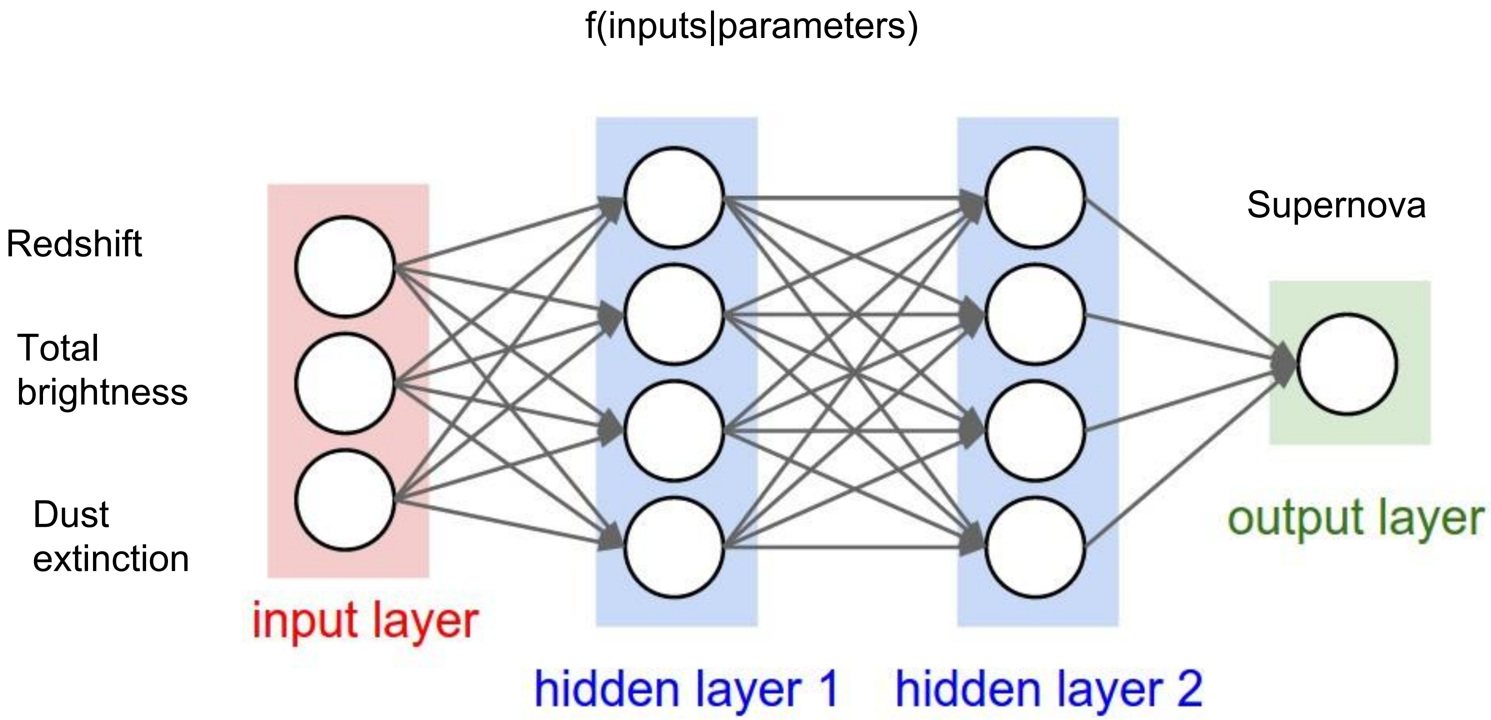
Total
brightness

Standard
Deviation



Prediction :
Eclipse

First attempt : Fully Connected Neural Network

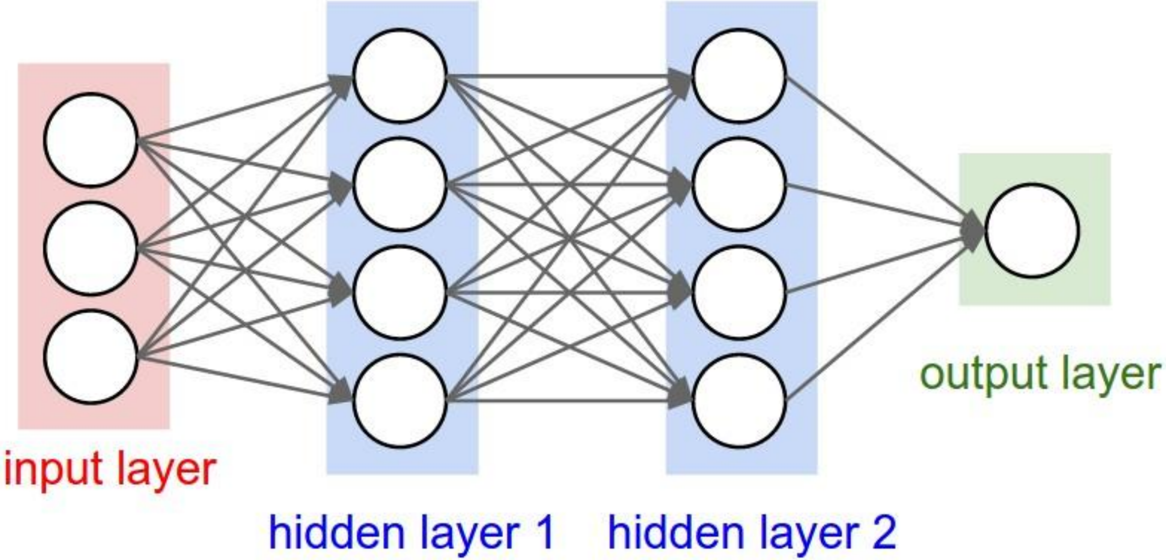


How do we feed in sequential data?

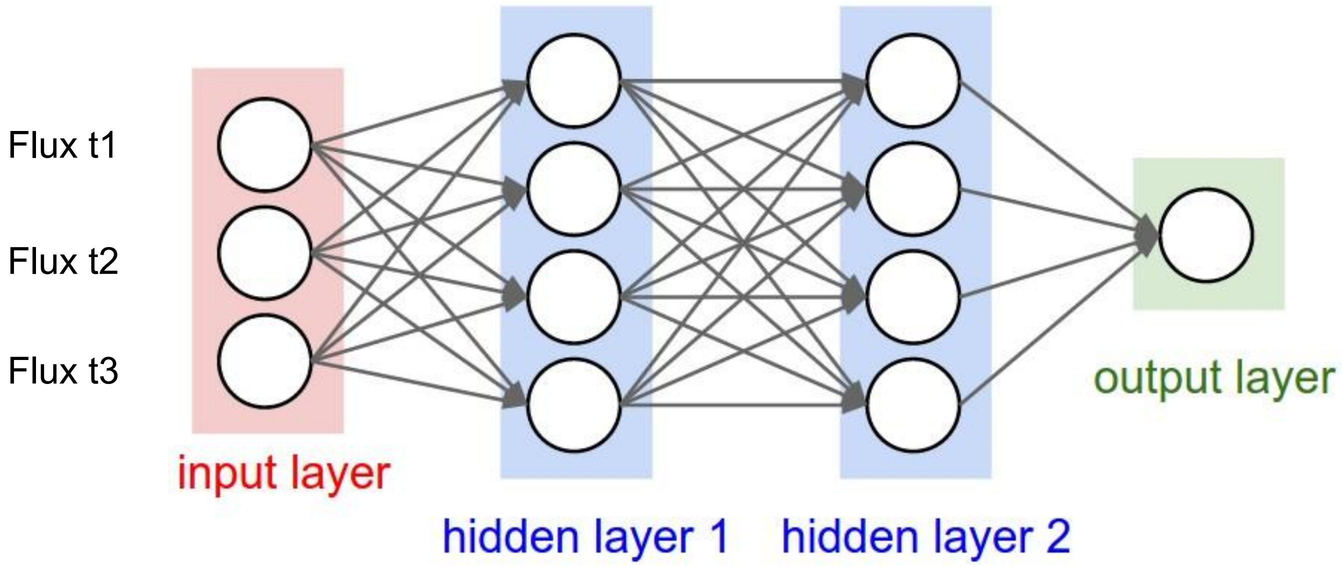
Flux t0

Flux t1

Flux t2

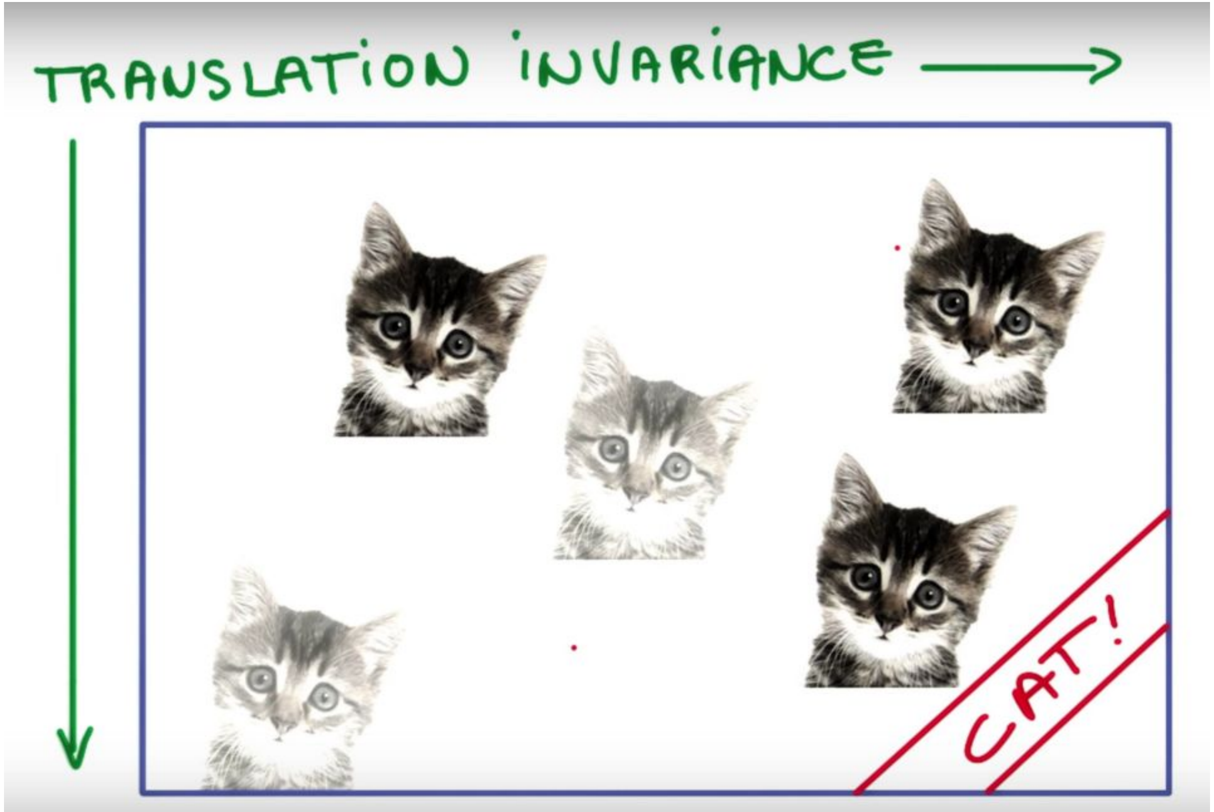


How do we feed in sequential data?



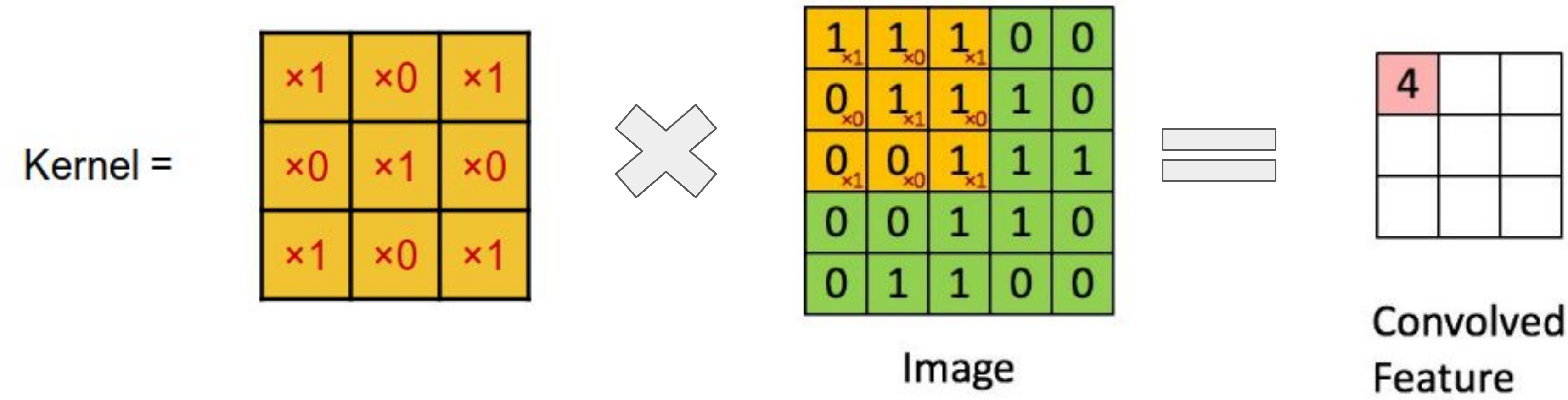
Need to relearn the rules at each point in the sequence !

Same problem with images and spatial translation



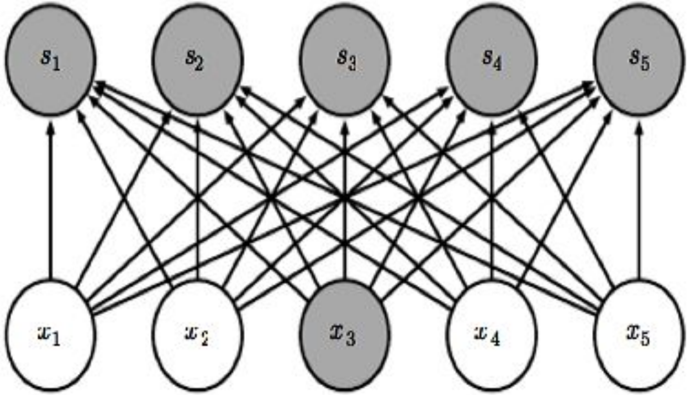
Solution: Convolutional network

Use convolutions instead of matrix multiplication.

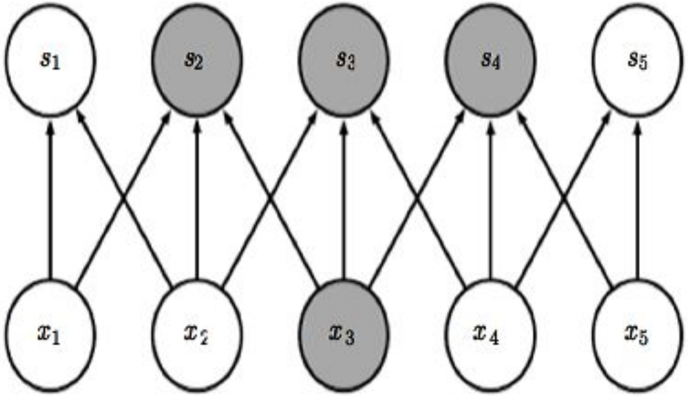


i) Sparse connections

- Often features can be detected in small patches of an image. We don't need to connect very far away pixels -> fewer parameters.



Fully Connected

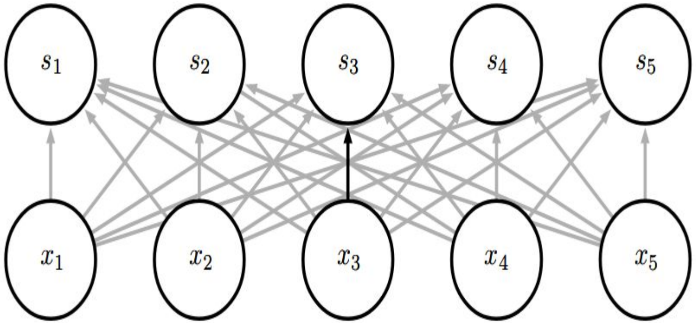


Sparse Connections

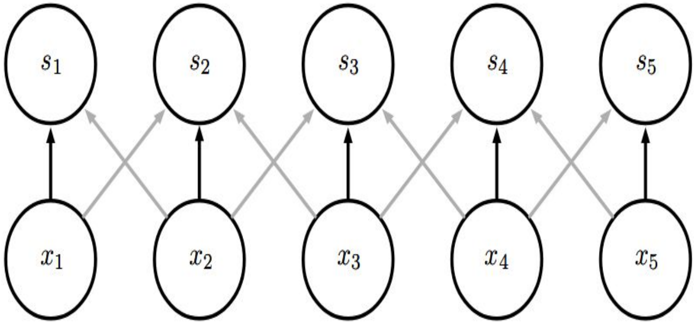
Deep Learning (2016), Ian Goodfellow, Yoshua Bengio and Aaron Courville

ii) Parameter sharing

- Apply the same weights to different pixels of the image -> Extract global features in an image.



Fully Connected



Shared parameters

Deep Learning (2016), Ian Goodfellow, Yoshua Bengio and Aaron Courville

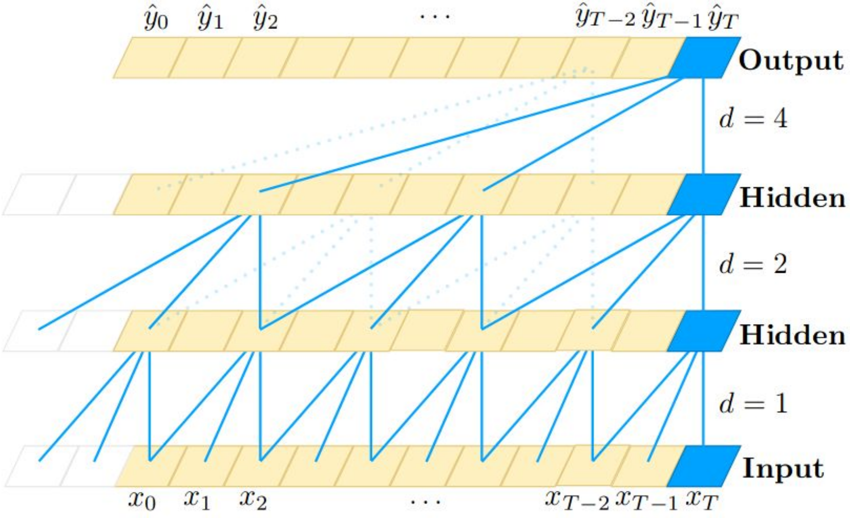
iii) Equivariant representation

- If the input changes the output changes in the same way. A function $f(x)$ is equivariant to a function g if:

$$f(g(x)) = g(f(x))$$

If we move an object in the input, its convolution moves in the same way in the output.

Second attempt: 1-D temporal convolution



Dilated Causal (at t, only see inputs no later than time t) Convolutions

arXiv:1803.01271

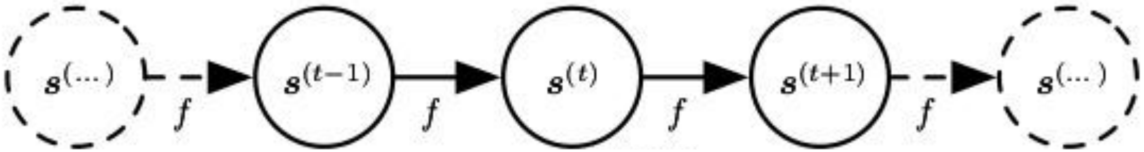
Cons

- Need a deep network to capture long-term dependencies. Partially solved by dilated convolutions (could also chose larger filter sizes)

- Problem with sparse connections, sometimes can’t capture the necessary long range dependencies.

Third attempt: Recurrent relation + Weight sharing

Output [t] = f(Output [t-1] | parameters)



Same function means same neural network with the same parameters.

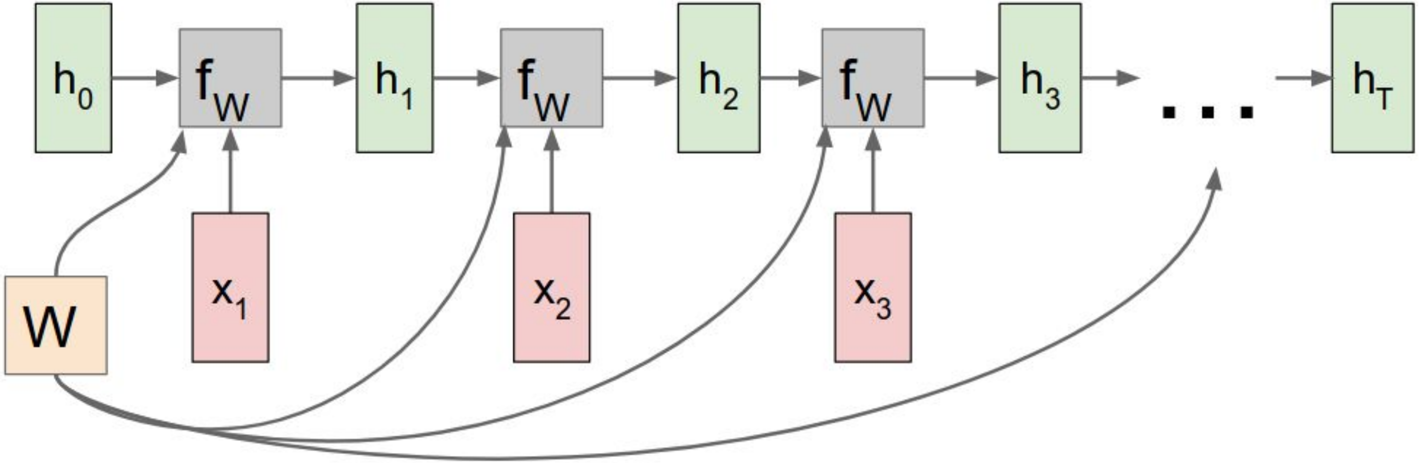
Are outputs the best way to relate previous knowledge with current input?

Predict next character : h + e + l + l + ?

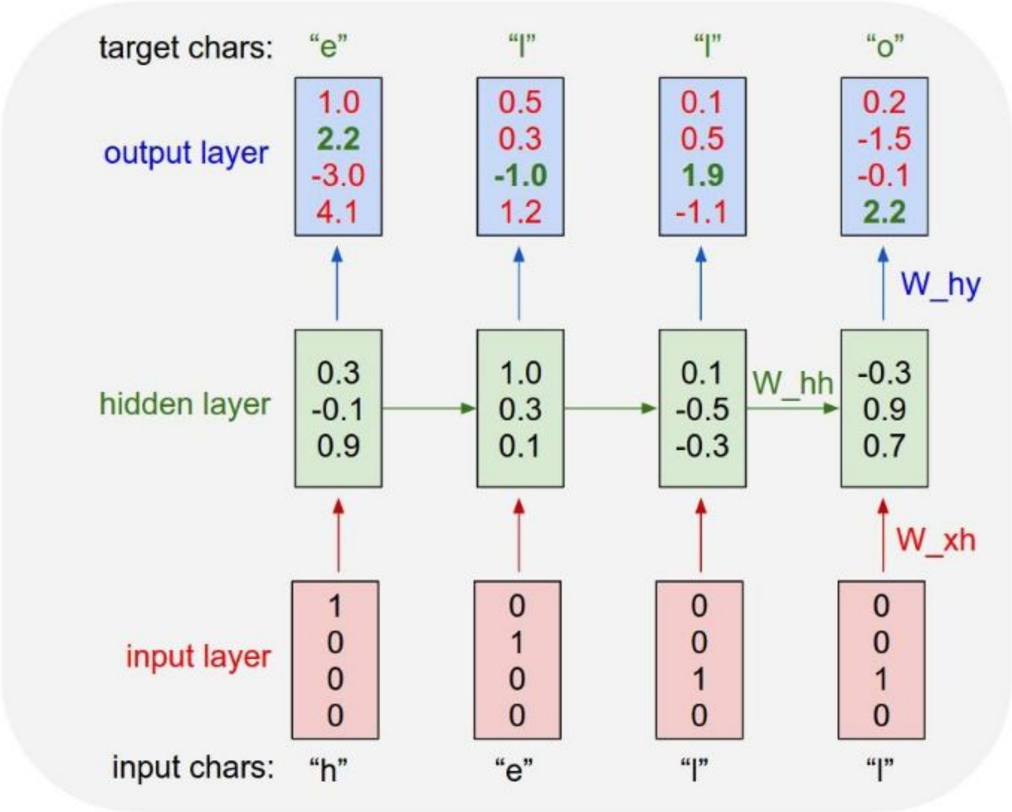
Deep Learning (2016), Ian Goodfellow, Yoshua Bengio and Aaron Courville

Using the previous state of the network

The state is a lossy reduced representation of what the network has seen before.

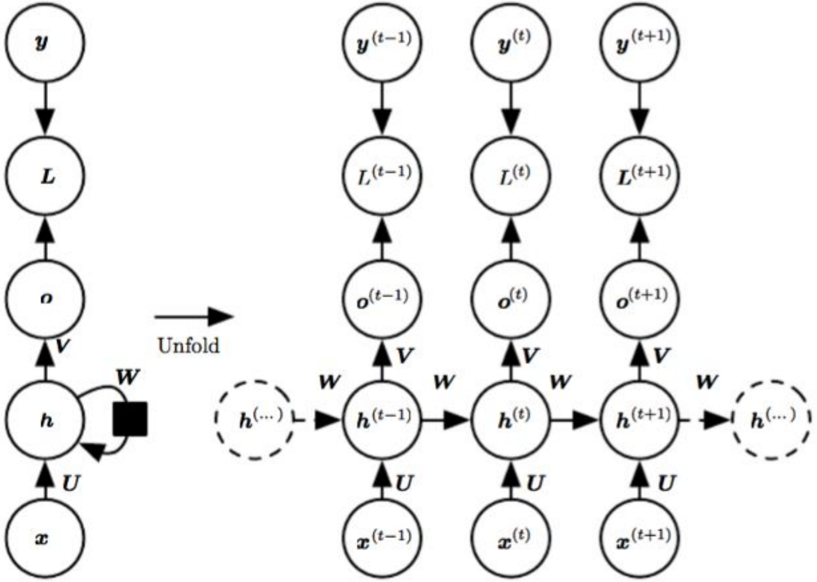


[Stanford lectures slides]



[Stanford lectures slides]

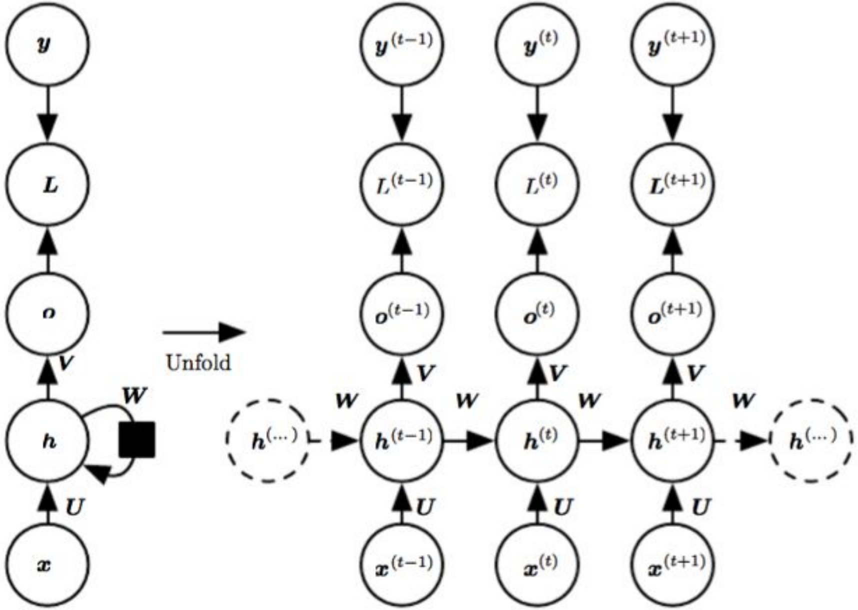
RNN: Recurrent Neural Networks



Weight sharing : the relation between previous time step and the next does not depend on time (stationary)

Deep Learning (2016), Ian Goodfellow, Yoshua Bengio and Aaron Courville

Forward pass



$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

Deep Learning (2016), Ian Goodfellow, Yoshua Bengio and Aaron Courville

Back Propagation Through Time (BPTT)

- Compute the mean loss across the different time steps.

- Back propagate the gradient of the loss respect to the shared weights over time.

- Problem: The gradient respect to the weights will involve products of the weight matrix -> Gradients could vanish, therefore no long term dependencies will be learned.

Vanishing gradients problem

Through the state recurrent relation:

$$\mathbf{h}^{(t)} = \mathbf{W}^T \mathbf{h}^{(t-1)} \rightarrow \mathbf{h}^{(t)} = \left(\mathbf{W}^t\right)^T \mathbf{h}^{(0)}$$

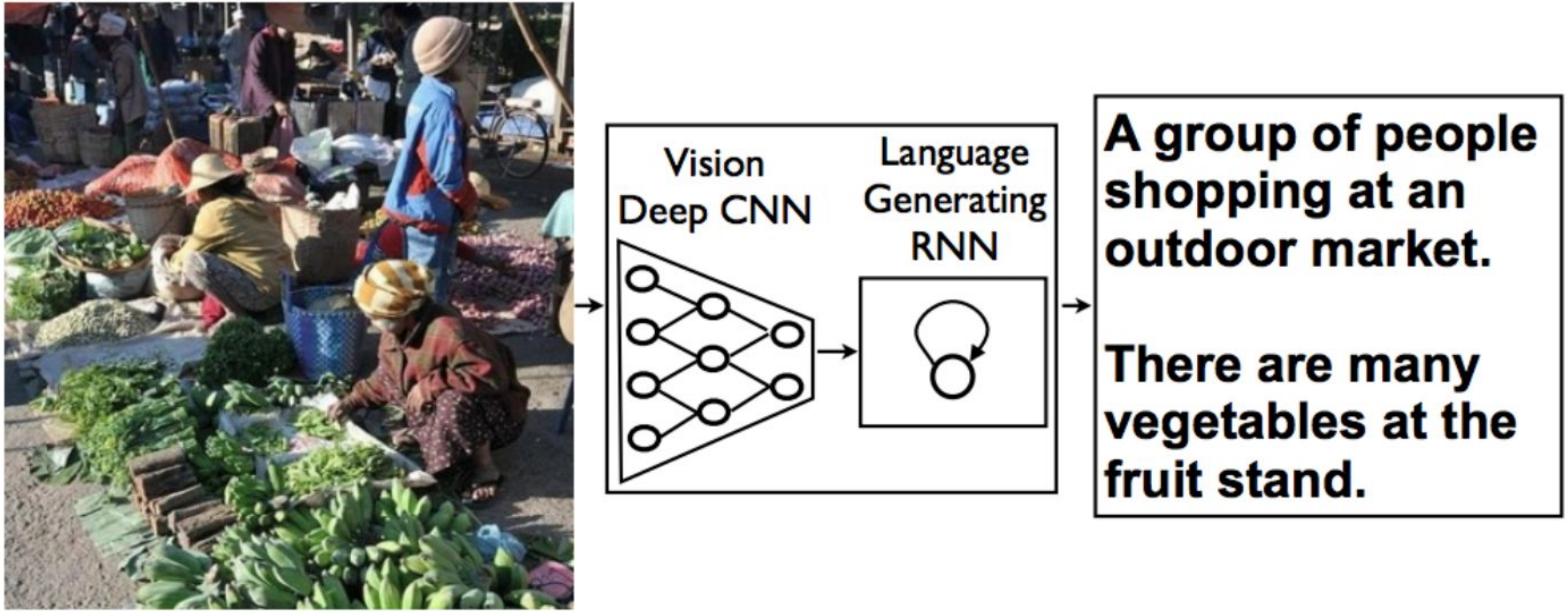
If we further decompose the weight matrix into its eigenvalues:

$$\mathbf{W} = \mathbf{Q}\mathbf{A}\mathbf{Q}^T$$

$$\mathbf{h}^{(t)} = \mathbf{Q}^T \mathbf{A}^t \mathbf{Q}\mathbf{h}^{(0)}$$

Eigenvalues smaller than 1 will decay to zero. Short term >> Long term !

Generating image captions



Conclusions

- Fully Connected networks can’t handle sequential data.

- Temporal Convolutional networks can, but we need to specify the range of the temporal dependency.

- Recurrent networks have a “memory” of what the network was doing previously.

- Theoretically, recurrent networks can handle long term dependencies. In practice, we find the vanishing gradients problem (LSTM as partial solution).