SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of AI &DS

Course Name – 19AD602 DEEP LEARNING

III Year / VI Semester

UNIT-4 OPTIMIZATION AND GENERALIZATION

Topic: Spatial Transformer Networks

Spatial Transformer Networks

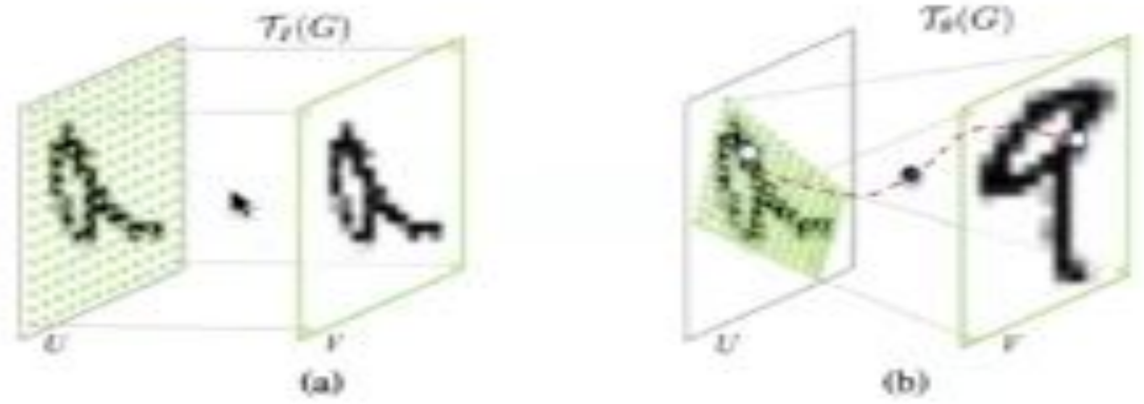
CASE STUDY:

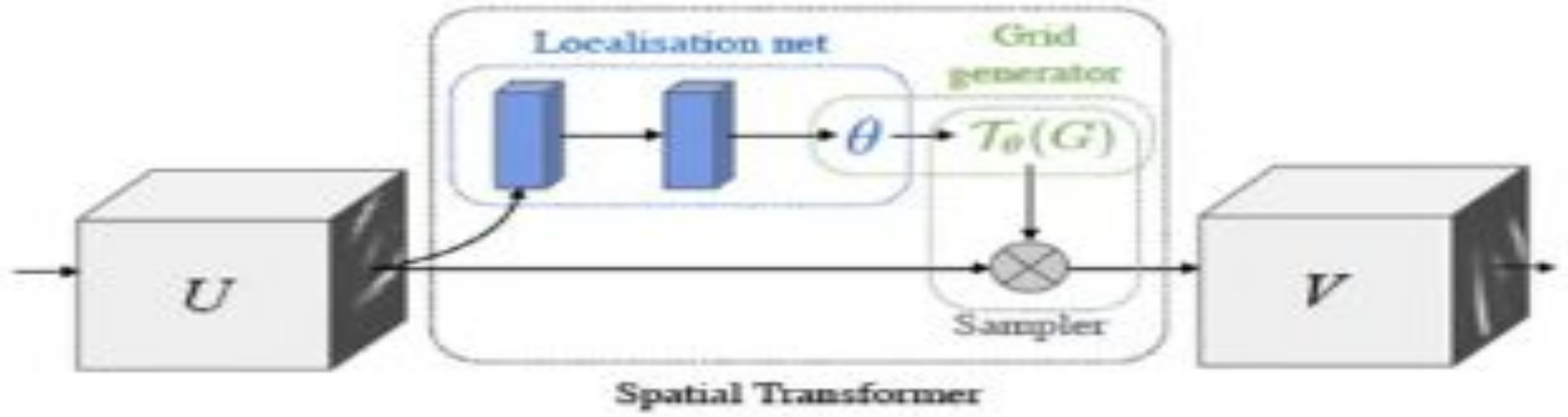
A financial firm used a Recurrent Neural Network (RNN) to predict stock prices based on historical time-series data. By training on past market trends, the RNN captured sequential dependencies, improving short-term forecasting accuracy. This helped traders make informed investment decisions, outperforming traditional statistical models.

Spatial Transformer Networks

Spatial transformer provides a space transformation that converts an input image to an output image with different shape. For example, applying the parameterized sampling grid to an image U produces the output V by the regular grid $G = \mathcal{T}_I(G)$, where I is the identity transformation parameters, or warping the regular grid with an affine transformation $G = \mathcal{T}_\theta(G)$ (translation, scale, rotation, and clutter).

A localization network inside a regular convolutional neural network is used to find the transformation parameters.

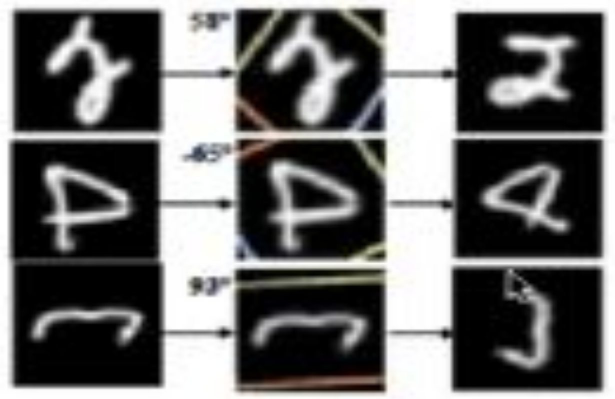


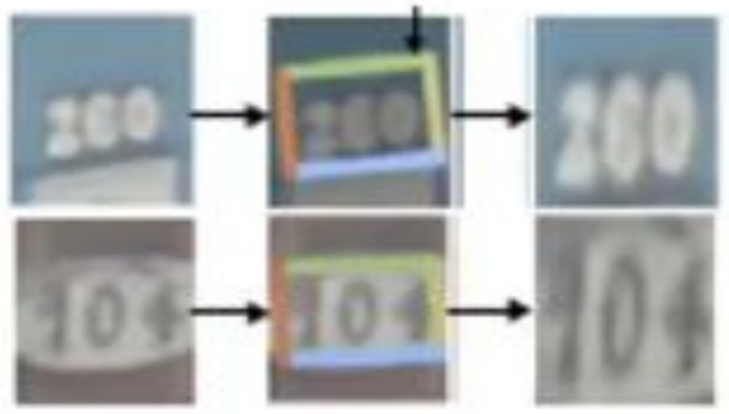


To perform a warping of the input feature map, each output pixel is computed by applying a sampling kernel centered at a particular location in the input feature map.

Spatial Transformer Networks

Spatial transformer consists of three models: localization network to generate spatial transformation parameter θ, and grid generator operates the predicted parameter θ on the input feature map and creates a sampling grid, and then the input feature map and the sampling grid are taken as inputs to the sampler, producing the output map sampled from the input at the grid points.





For a 2D affine transformation, $\mathcal{T}_\theta(G_i)$ is expressed as

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

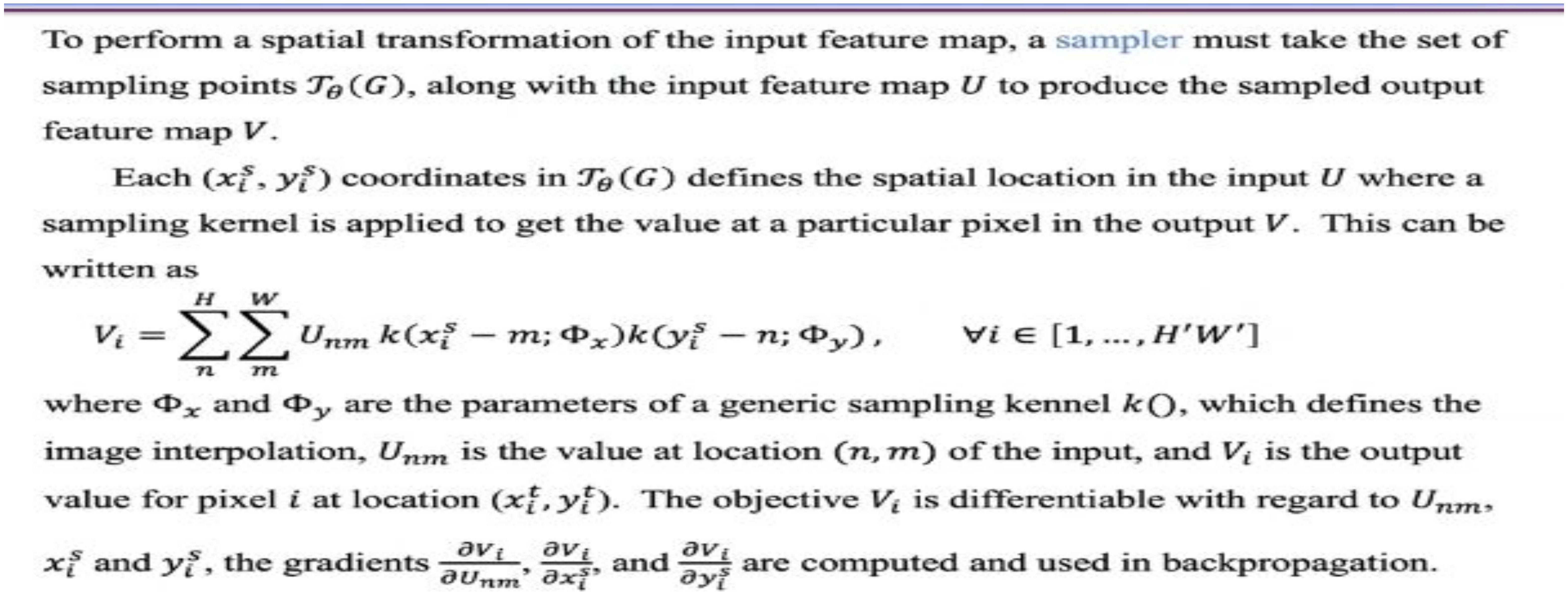
x_i^s and y_i^s: coordinates of source pixel

x_i^t and y_i^t: coordinates of target pixel

where A_θ is the affine transformation matrix.

Spatial Transformer Networks

DIFFERENT SAMPLING :

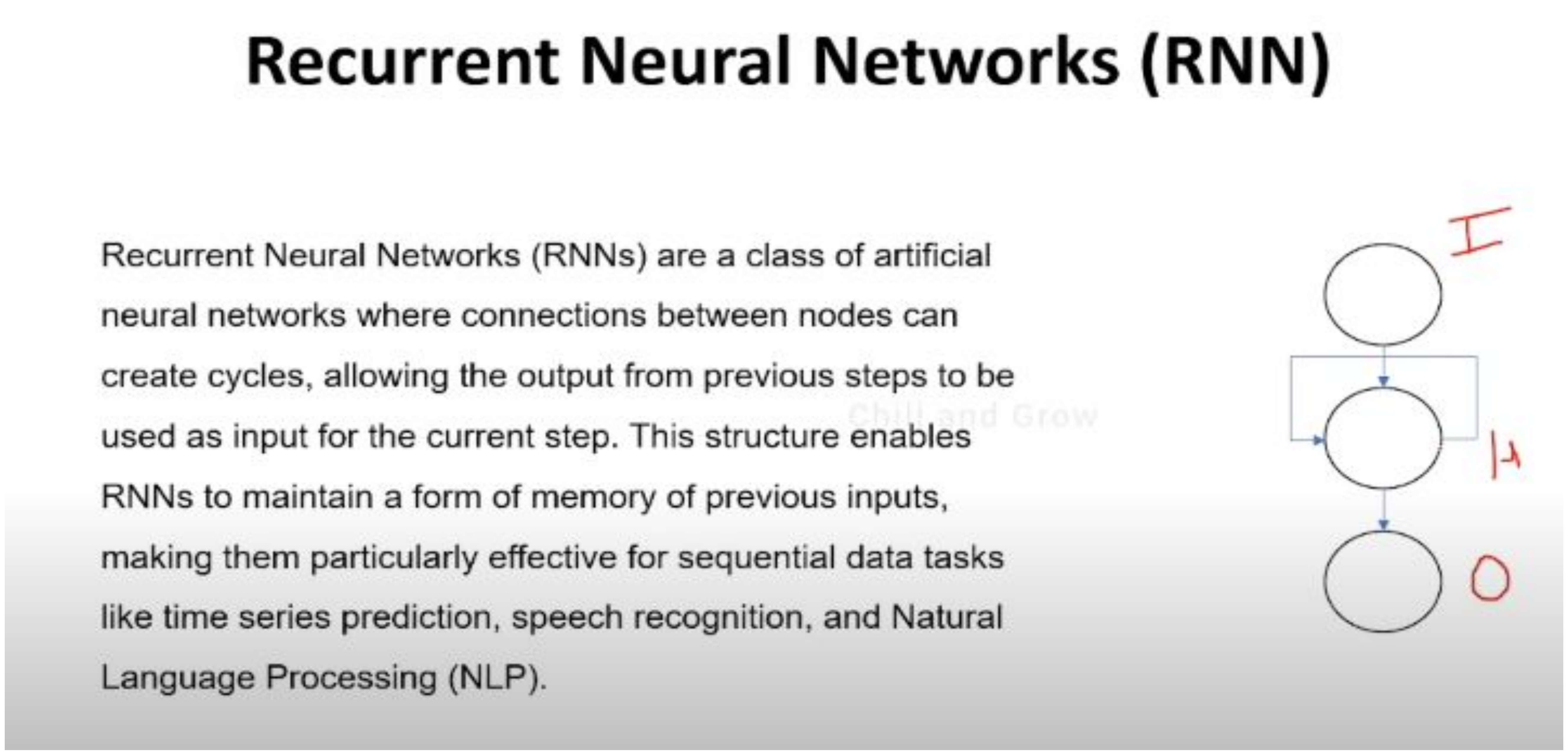

To perform a spatial transformation of the input feature map, a sampler must take the set of sampling points $\mathcal{T}_\theta(G)$, along with the input feature map U to produce the sampled output feature map V.

Each (x_i^s, y_i^s) coordinates in $\mathcal{T}_\theta(G)$ defines the spatial location in the input U where a sampling kernel is applied to get the value at a particular pixel in the output V. This can be written as

$$V_i = \sum_n^H \sum_m^W U_{nm}\, k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y), \qquad \forall i \in [1, \dots, H'W']$$

where Φ_x and Φ_y are the parameters of a generic sampling kennel $k()$, which defines the image interpolation, U_{nm} is the value at location (n, m) of the input, and V_i is the output value for pixel i at location (x_i^t, y_i^t). The objective V_i is differentiable with regard to U_{nm}, x_i^s and y_i^s, the gradients $\frac{\partial V_i}{\partial U_{nm}}, \frac{\partial V_i}{\partial x_i^s}$, and $\frac{\partial V_i}{\partial y_i^s}$ are computed and used in backpropagation.

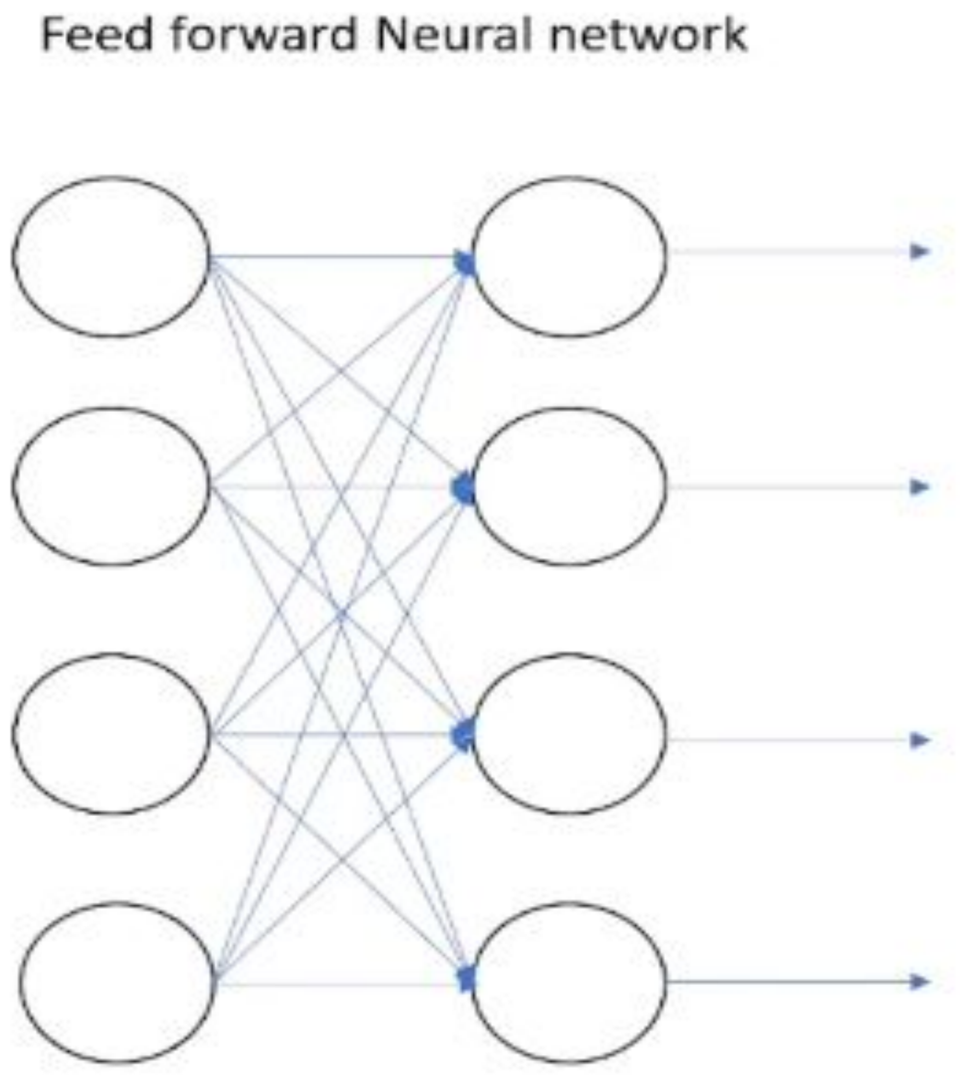
RNN Networks

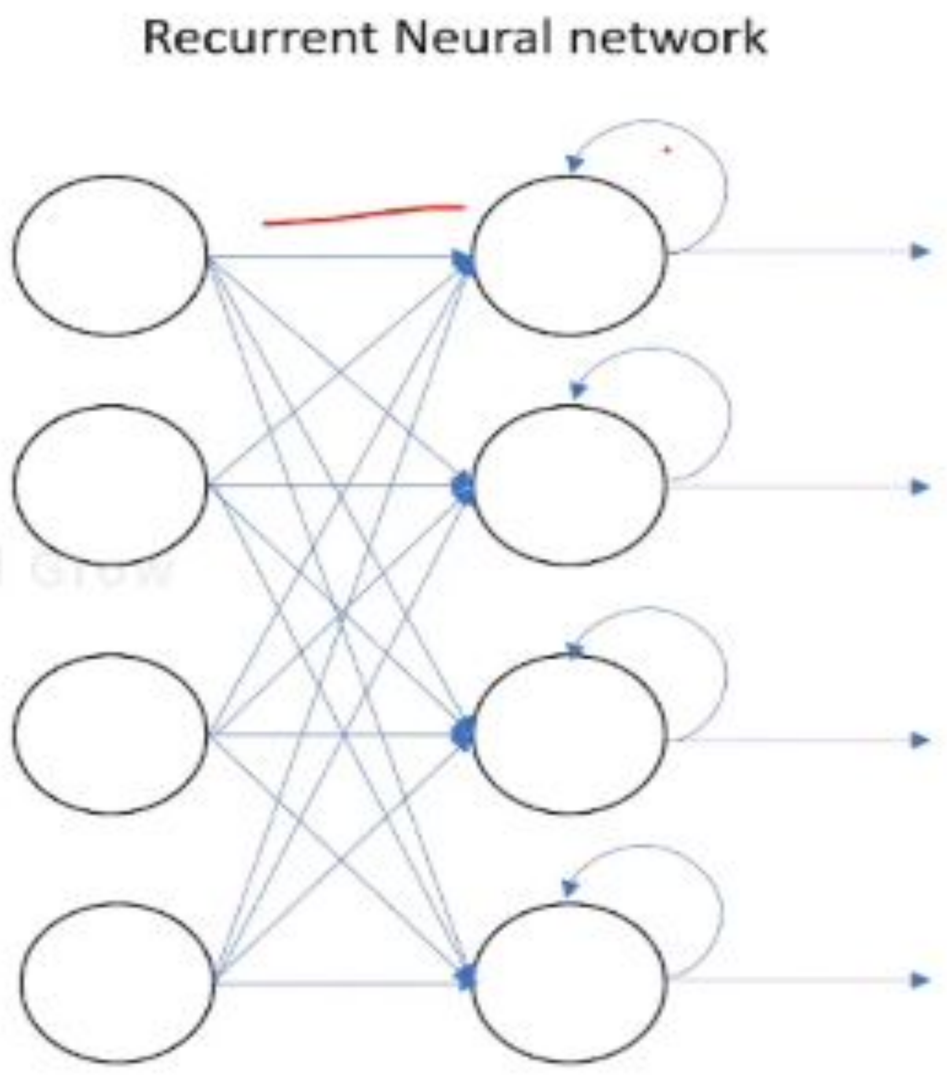


Recurrent Neural Networks (RNN)

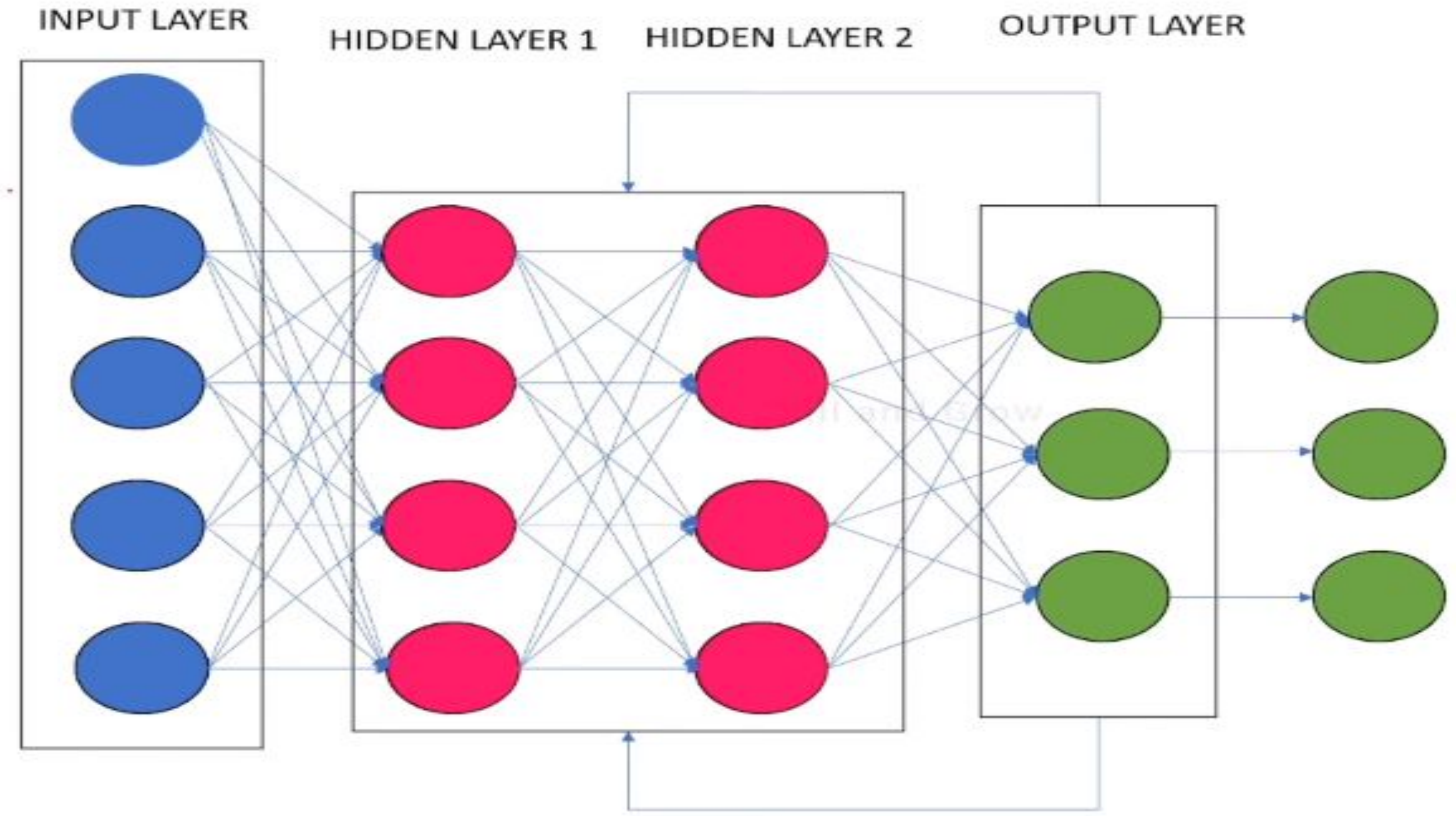
Recurrent Neural Networks (RNNs) are a class of artificial neural networks where connections between nodes can create cycles, allowing the output from previous steps to be used as input for the current step. This structure enables RNNs to maintain a form of memory of previous inputs, making them particularly effective for sequential data tasks like time series prediction, speech recognition, and Natural Language Processing (NLP).

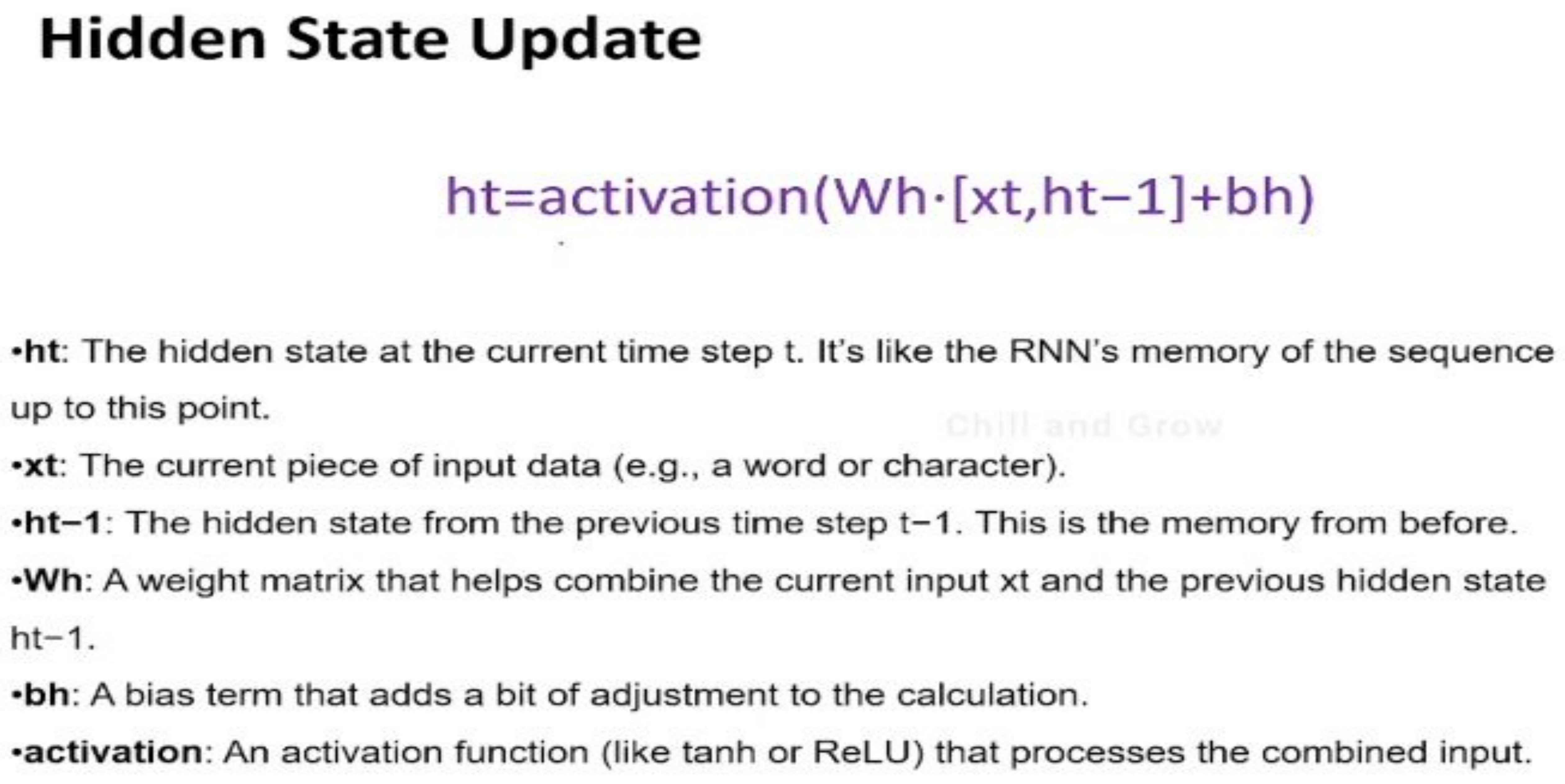
RNN Networks





RNN Networks



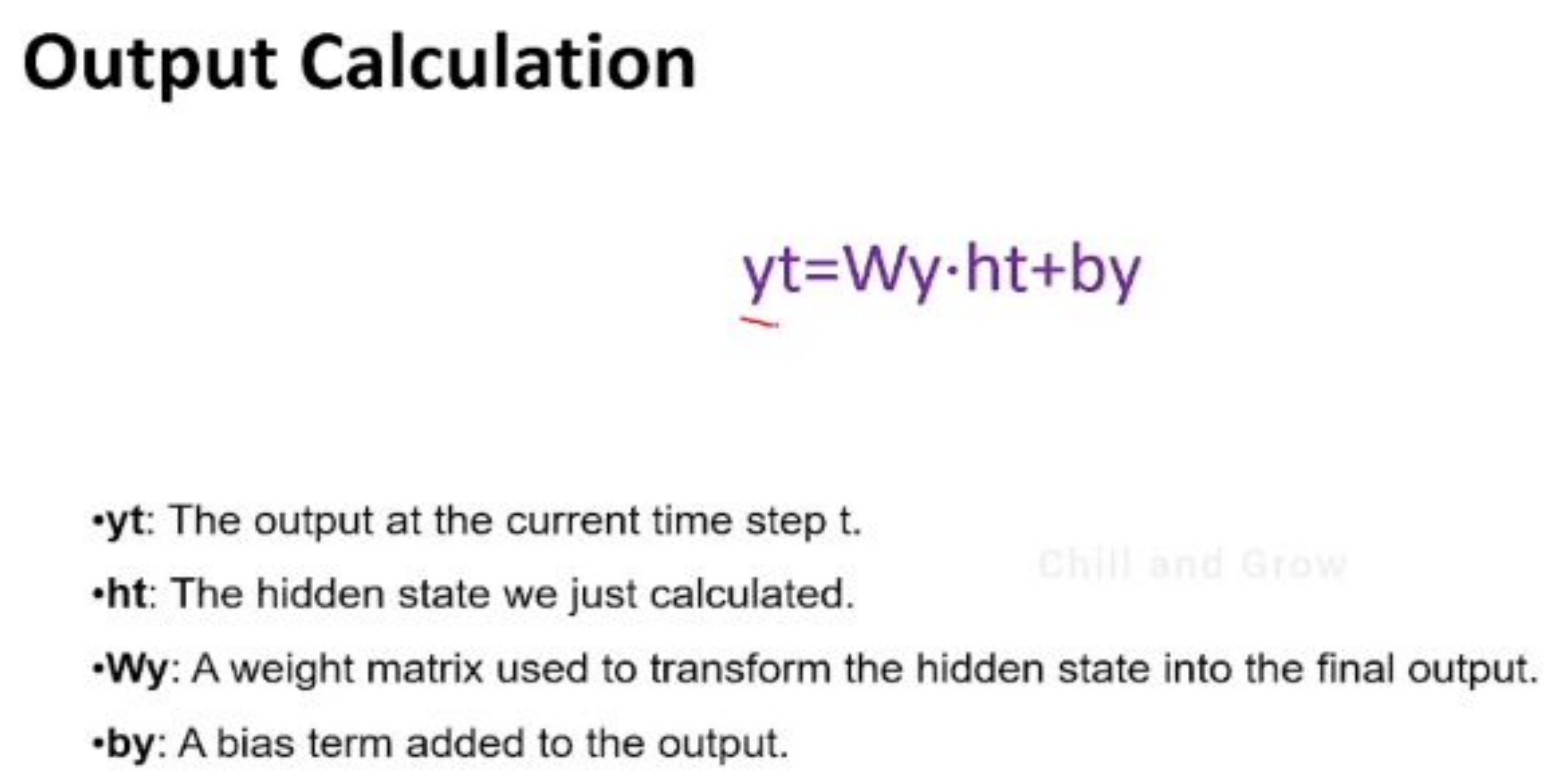


Hidden State Update

$$ht=activation(Wh\cdot[xt,ht-1]+bh)$$

- **ht**: The hidden state at the current time step t. It's like the RNN's memory of the sequence up to this point.
- **xt**: The current piece of input data (e.g., a word or character).
- **ht−1**: The hidden state from the previous time step t−1. This is the memory from before.
- **Wh**: A weight matrix that helps combine the current input xt and the previous hidden state ht−1.
- **bh**: A bias term that adds a bit of adjustment to the calculation.
- **activation**: An activation function (like tanh or ReLU) that processes the combined input.

RNN Networks

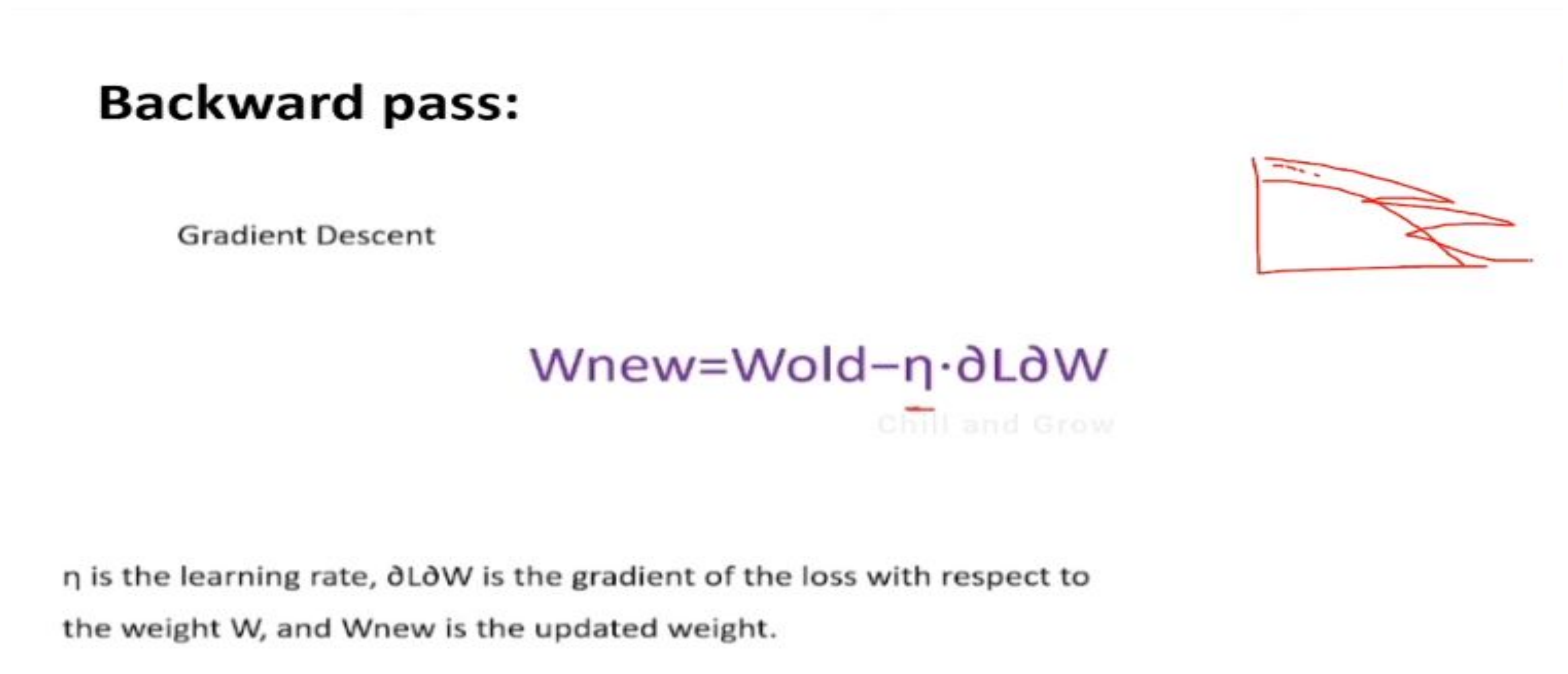


Output Calculation

$$yt=Wy\cdot ht+by$$

- **yt**: The output at the current time step t.
- **ht**: The hidden state we just calculated.
- **Wy**: A weight matrix used to transform the hidden state into the final output.
- **by**: A bias term added to the output.

RNN Networks



Backward pass:

Gradient Descent

$$Wnew=Wold-\eta\cdot\partial L\partial W$$

η is the learning rate, $\partial L\partial W$ is the gradient of the loss with respect to the weight W, and Wnew is the updated weight.

RNN Networks

TYPES:

- ONE TO ONE
- ONE TO MANY
- MANY TO ONE
- MANY TO MANY

VARIENTS:

- VANILLA
- BIDIRECTIONAL
- LSTM
- GRU

RNN Networks

THANK YOU