
13.11.2024

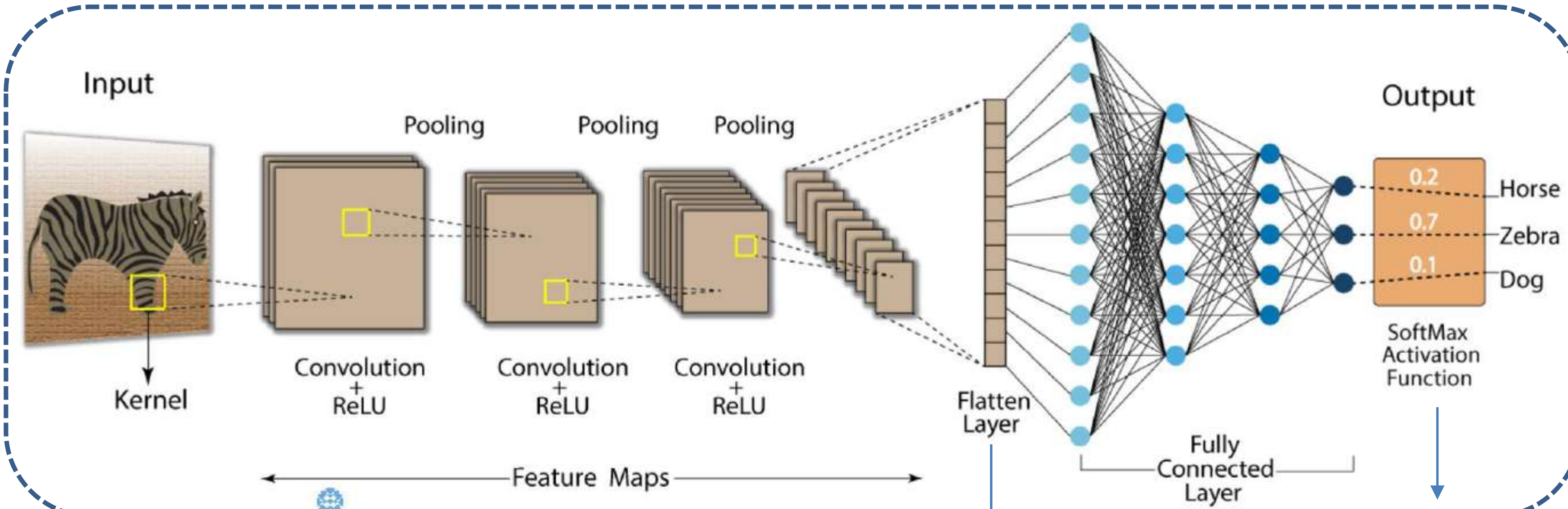
# BITS F464: Machine Learning (1st Sem 2024-25)

## DEEP LEARNING MODELS: CNN, RNN, LSTM, GRU, GAN

Chittaranjan Hota, Sr. Professor
Dept. of Computer Sc. and Information Systems
hota@hyderabad.bits-pilani.ac.in

# Convolutional Neural Networks: Deep Learning

- So far…classified real values or discrete categories. What about Images & Sequences?
- Multi-layer Perceptrons (MLPs) are generally fully connected (each neuron in one layer is connected to every neuron in the subsequent layer).
- If Input: M units and Output N units, we need MXN connections. For an input image M of 256X256 = 65563 grayscale pixels, and output N of 1000 units, we would need 65 million connections.
- In Image data: We might want 'Share structure property" and "Invariance" property to be encoded into the NN's architecture. CNNs to rescue.

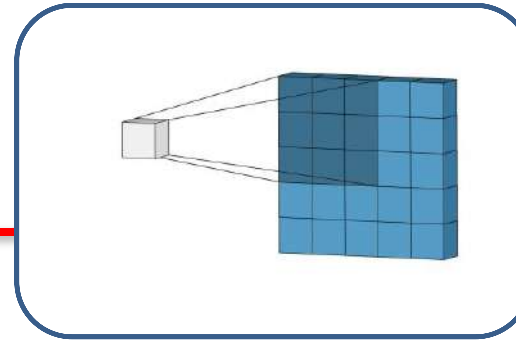



(Img. Source: https://developersbreach.com/)

Spatial structure to 1-D vector

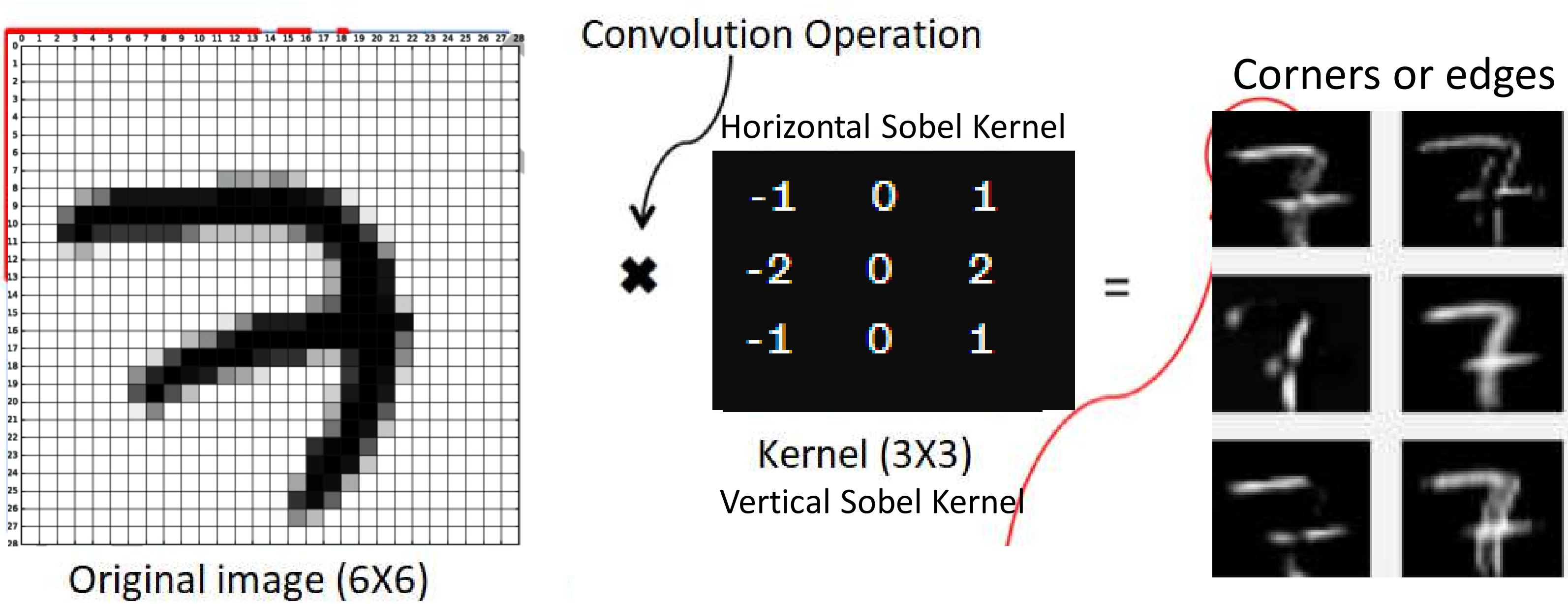
$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)}$$

# CNN: Convolution Layer

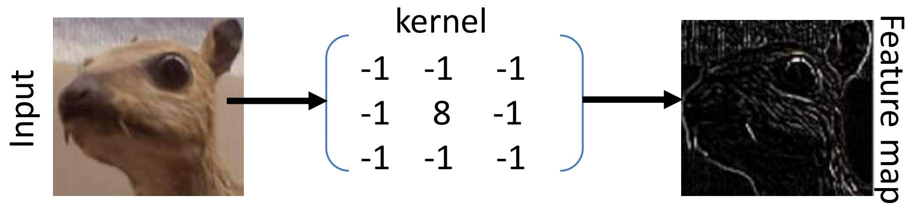


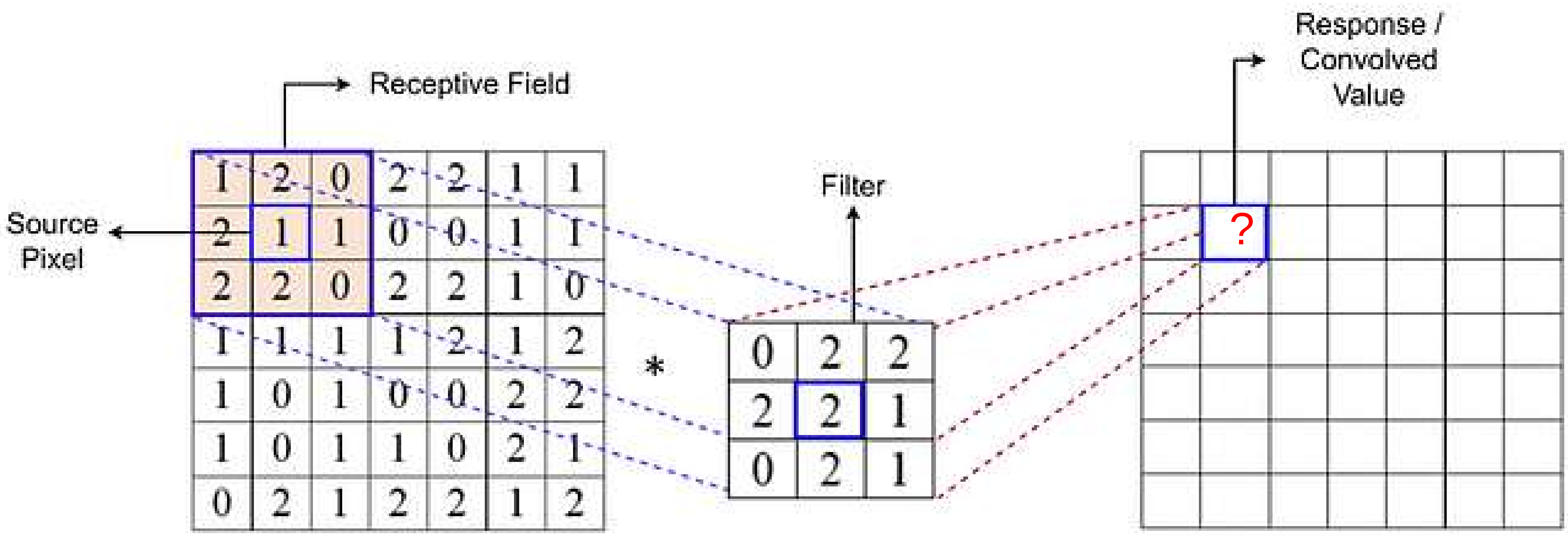
- What is the need of Convolution Operation?
  - To extract features from input data by applying a kernel (also known as a filter) over the input.
  - When a convolutional layer is applied to an input image, the resulting feature maps often have smaller spatial dimensions compared to the input.

Dot product: $(X_*W)\,[i, j] = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} X[m,n]\cdot W[i-m, j-n]$

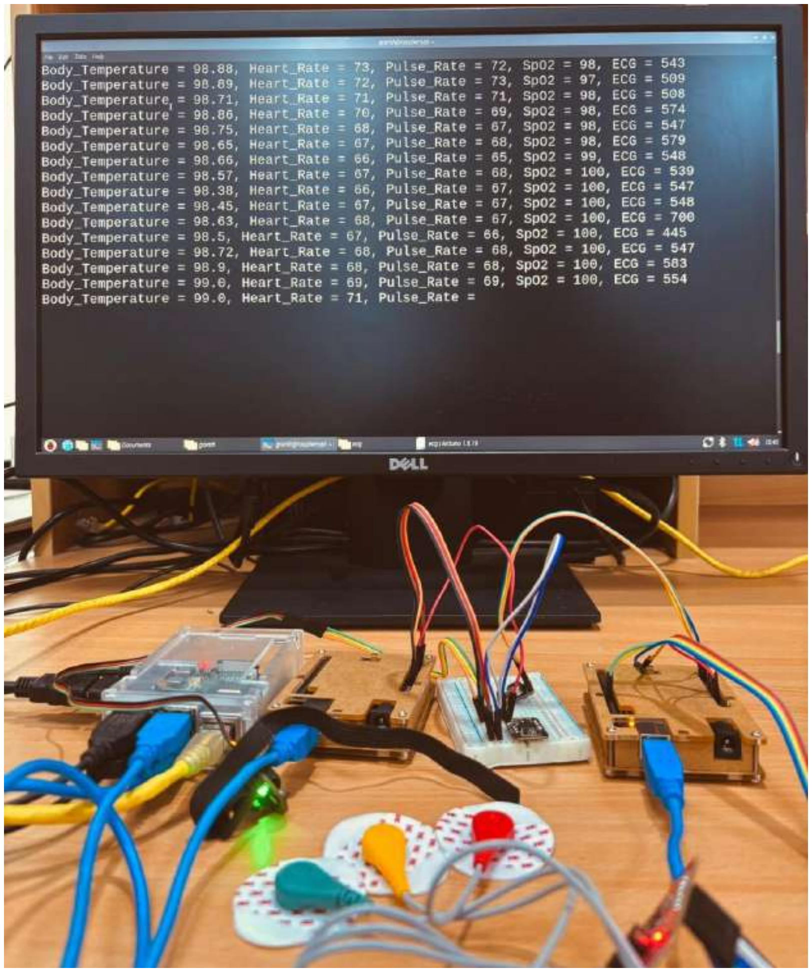


# Convolution in CNNs continued…



# CNN uses at our Lab (SmartNet AI Lab)




**File Uploaded Successfully**
test-data.csv

Analyzing WBAN Sensor Data from file: test-data.csv

Anomaly Type

The dataset provide is an instance of Context Anomaly

Why do you think this dataset is leading to a context anomaly and what disease might be causing it

The provided dataset is interesting because it shows some potential anomalies, particularly in the Heart Rate and ECG readings. While the Body Temperature and SpO2 levels remain within a typical range, the Heart Rate is fluctuating between the lower and higher ends of a normal range, and the ECG values are consistently above the expected range. This suggests there might be a potential issue with the heart's electrical activity. **Possible Causes of Anomaly:**

**Cardiac arrhythmias:** These are irregular heartbeats, which can manifest as fluctuations in heart rate and abnormalities in ECG readings.

**Stress or anxiety:** These can temporarily increase heart rate and potentially affect ECG patterns.

**Medication side effects:** Certain medications can influence heart rate and rhythm.

**Underlying medical conditions:** Other health conditions, such as thyroid disorders or electrolyte imbalances, can impact heart function and lead to changes in heart rate and ECG readings. **Important Note:** This is just an interpretation based on the limited information available. It's crucial to consult a medical professional for an accurate diagnosis based on a comprehensive evaluation and further testing.

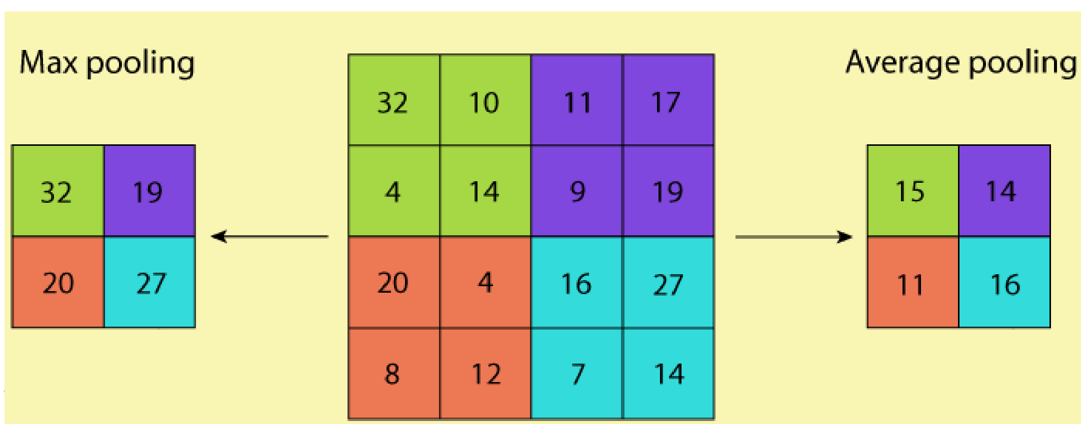
# CNN: Pooling (or Sub-sampling) Layer

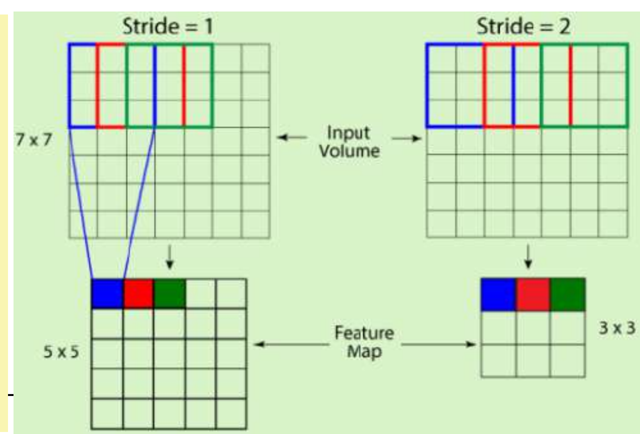
**Spatial Invariance**: Pooling layers aggregate information from local neighborhoods of the input feature map, which helps in creating spatial invariance.

→ Spatial variations in the input (such as translation, rotation, or scaling) are tolerated to some extent, making the network more robust to variations in input data.

**Dimensionality Reduction:** downsamples the feature maps while retaining the important features. This reduces the number of parameters (weights) reducing the chances of Overfitting.

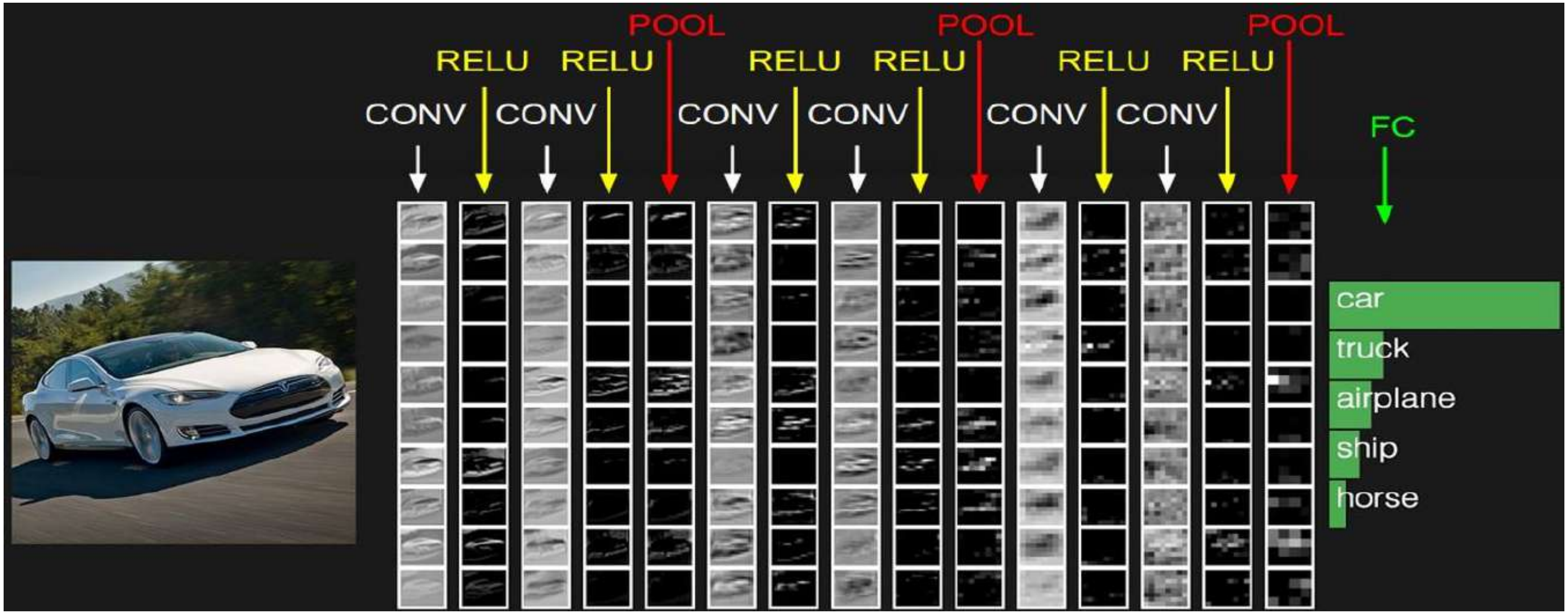
**Local feature detection:** Salient features within the local neighborhood is identified.



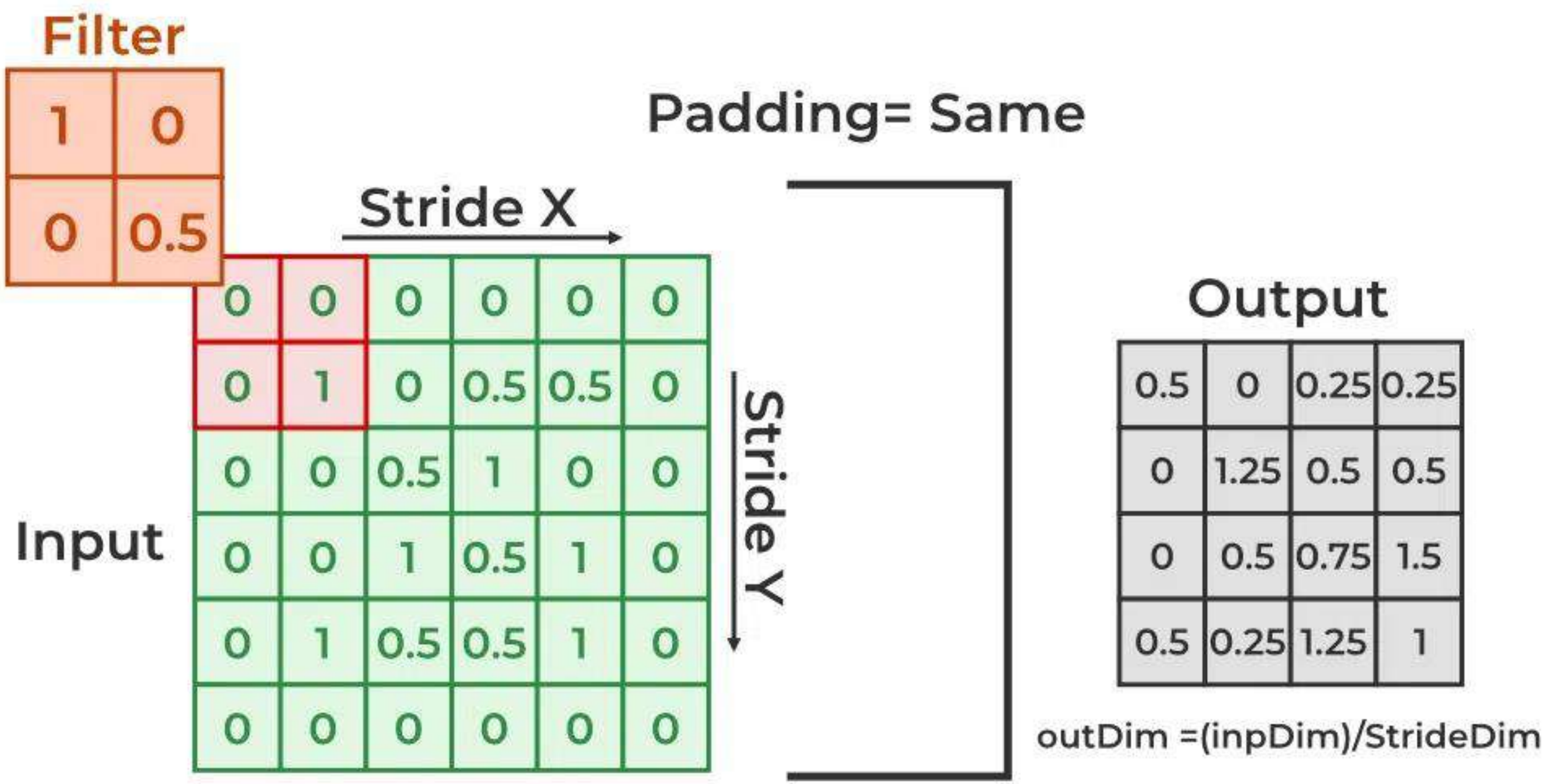

(Img. Source: https://developersbreach.com/)

# Classifying an Image: An example




Acknowledgement: Md Mahin's presentation.

# CNN: Padding

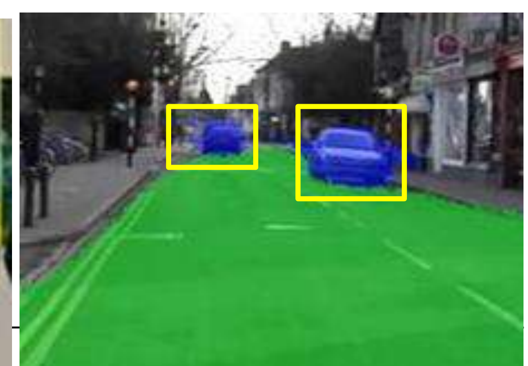



- It controls the spatial dimensions of feature maps and prevents information loss at the borders of the image.

Img. Source: geeksforgeeks

# Other types of Convolutional Networks: FCN, SSMD

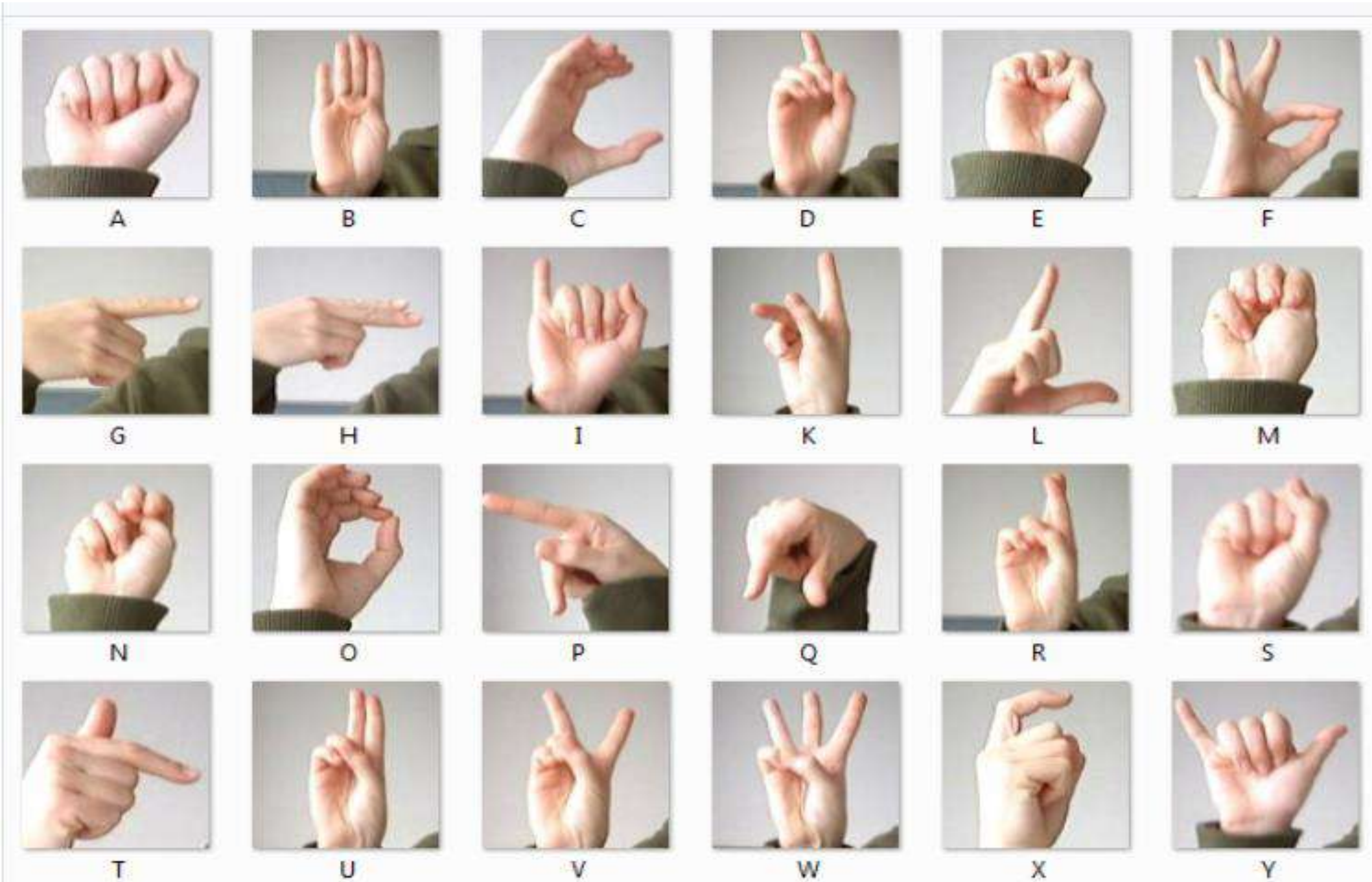
- Semantic segmentation, where the goal is to assign a class label to each pixel in the input image, the fully connected layer is not well-suited as they do NOT preserve spatial information.
- In Fully Convolutional Networks (FCNs), the final fully connected layer of the traditional CNN is replaced with **convolutional layers**. They allow the network to produce an output feature map with the same spatial dimensions as that of input.
- Some Object Detection Networks: In object detection architectures like YOLO (You Only Look Once) or SSD (Single Shot Multibox Detector), fully connected layers are often replaced by convolutional layers with spatial dimensions reduced to 1x1. Making network to predict bounding boxes and class probabilities at different spatial locations in the image.

Img. Source: https://www.mathworks.com/

# Mini-Project on hand gesture recognition using CNNs

Similar to LeNet

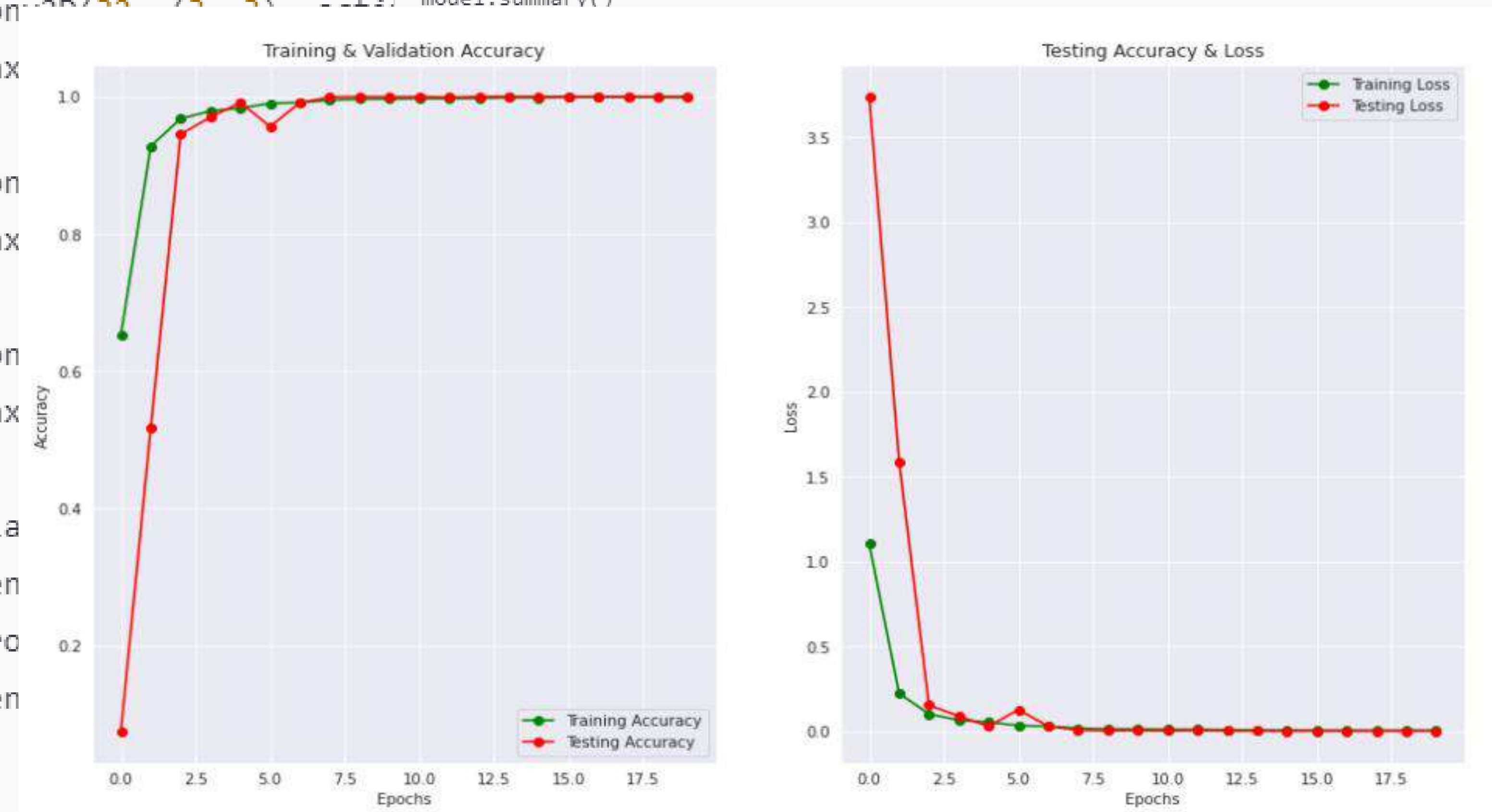



Assignment 5: Submission deadline: 23.11.2024 Many others: ResNet, AlexNet, ImageNet

# Mini-Project Continued…

```
# Define the CNN model
model = Sequential([
    Con
    Max

    Con
    Max

    Con
    Max

    Fla
    Den
    Dro
    Den
])
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Display the model architecture
model.summary()

rue)
```

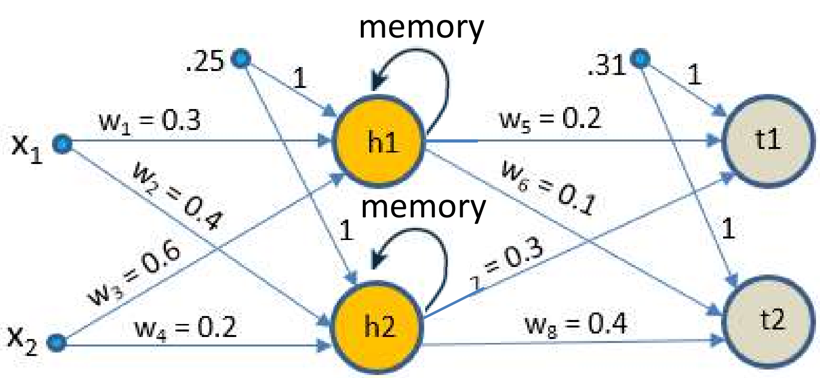



Submission deadline: 23.11.2024

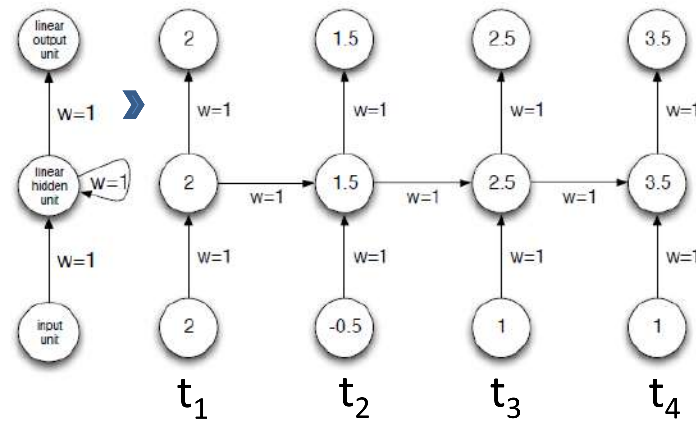
# Recurrent Neural Networks (RNNs)

- Feed Forward Neural Networks are Acyclic where data passes from input to the output nodes and not vice versa.
  - Once the FFNN is trained, its state is fixed and does not alter as new data is presented to it. It does not have memory.

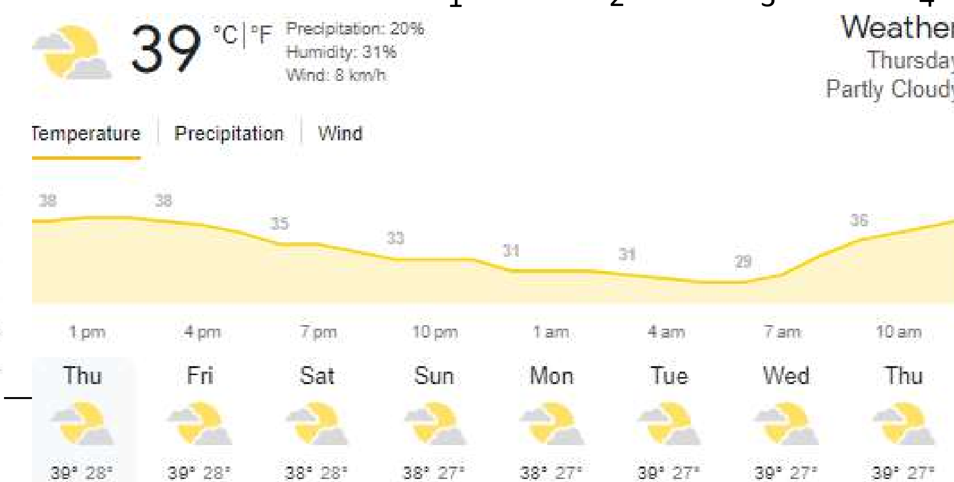
R
N
N




f
u
n
?




Applications

# RNN Applications

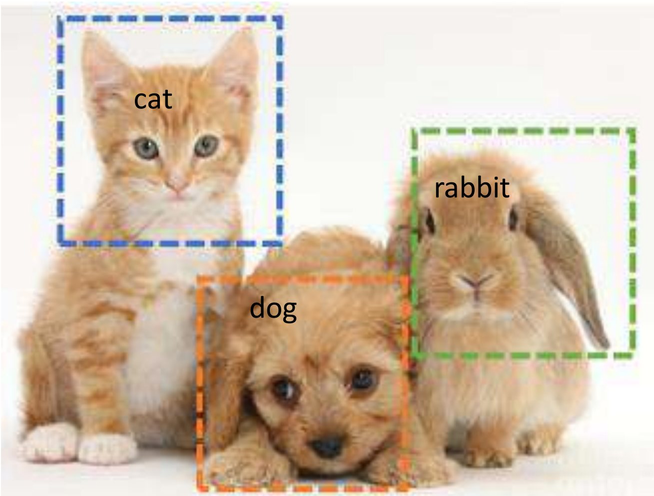



Image Classification

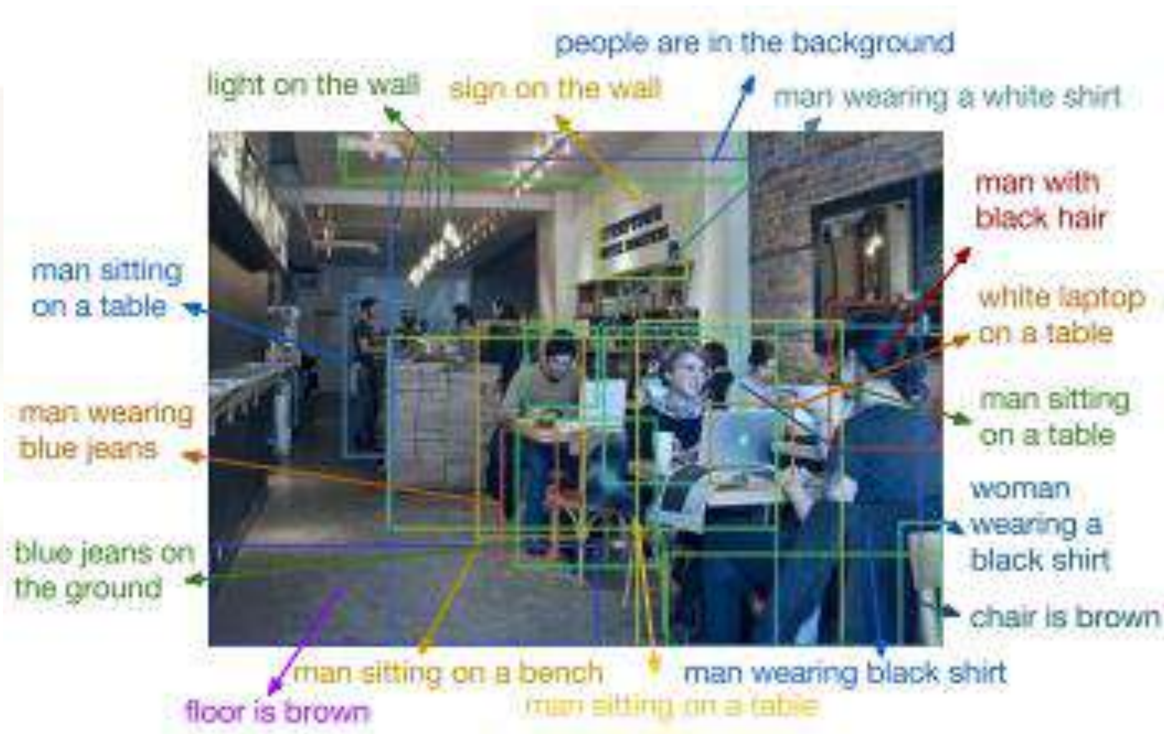
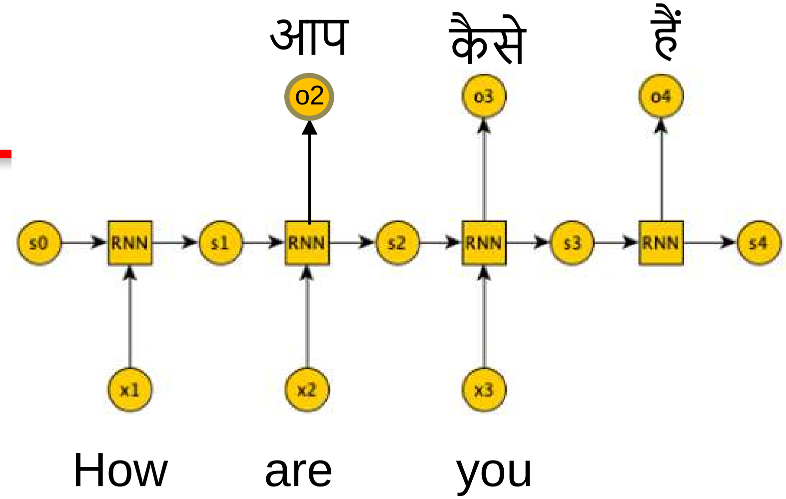
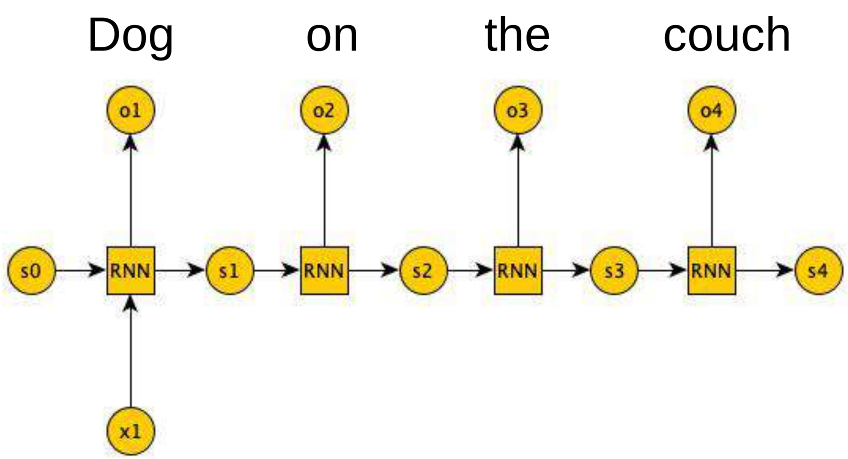
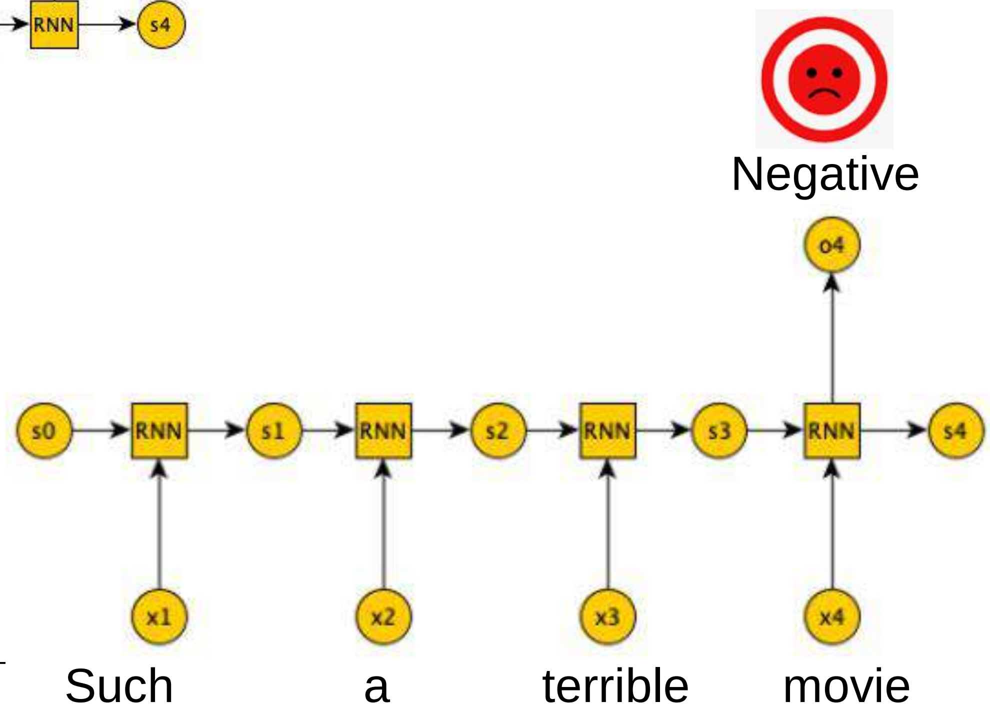



Image Captioning

# Example Usages…

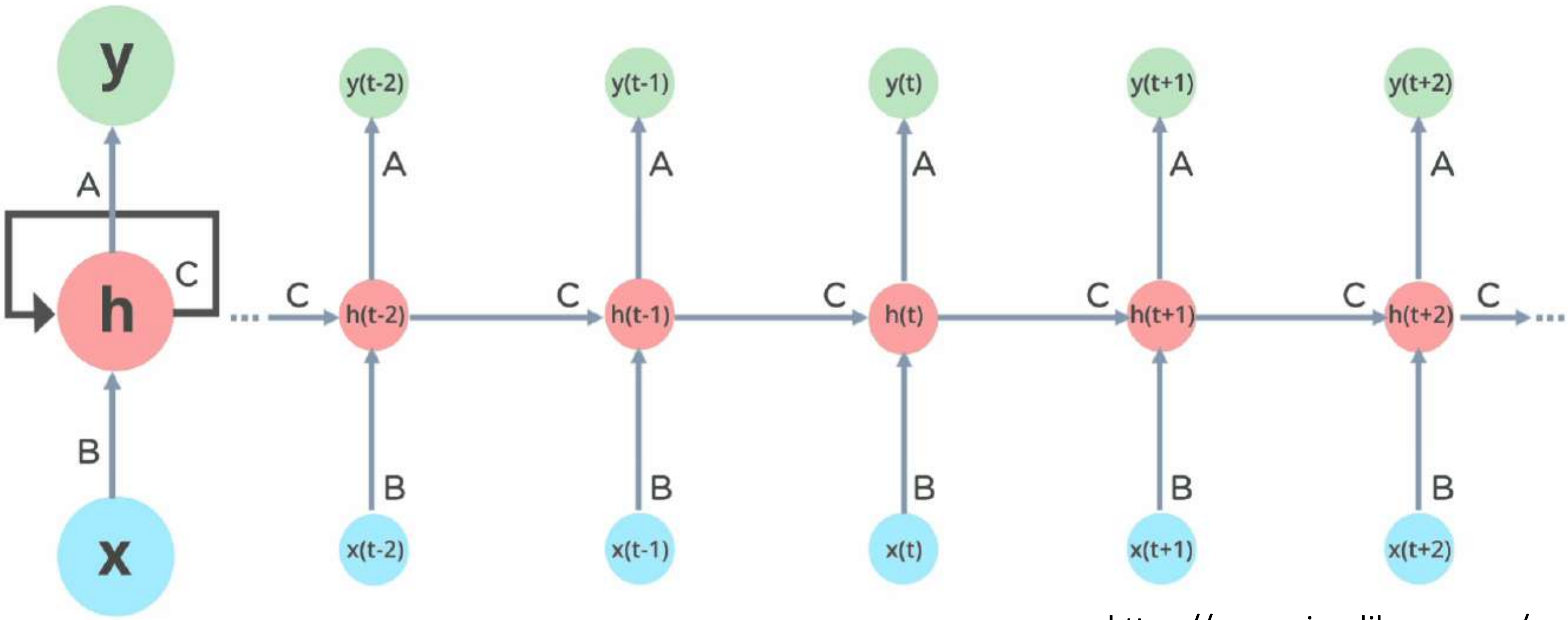


# Modelling sequential data: RNN

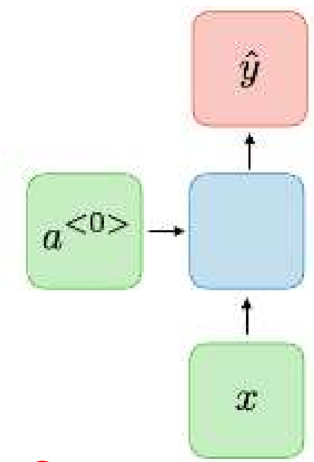
- A hidden state captures information about previous inputs. State is plugged back into itself.
- The hidden state is updated recurrently based on the current input and the previous hidden state.




https://www.simplilearn.com/

$g_1$ and $g_2$: Activation functions: Sigmoid or Tanh or ReLU

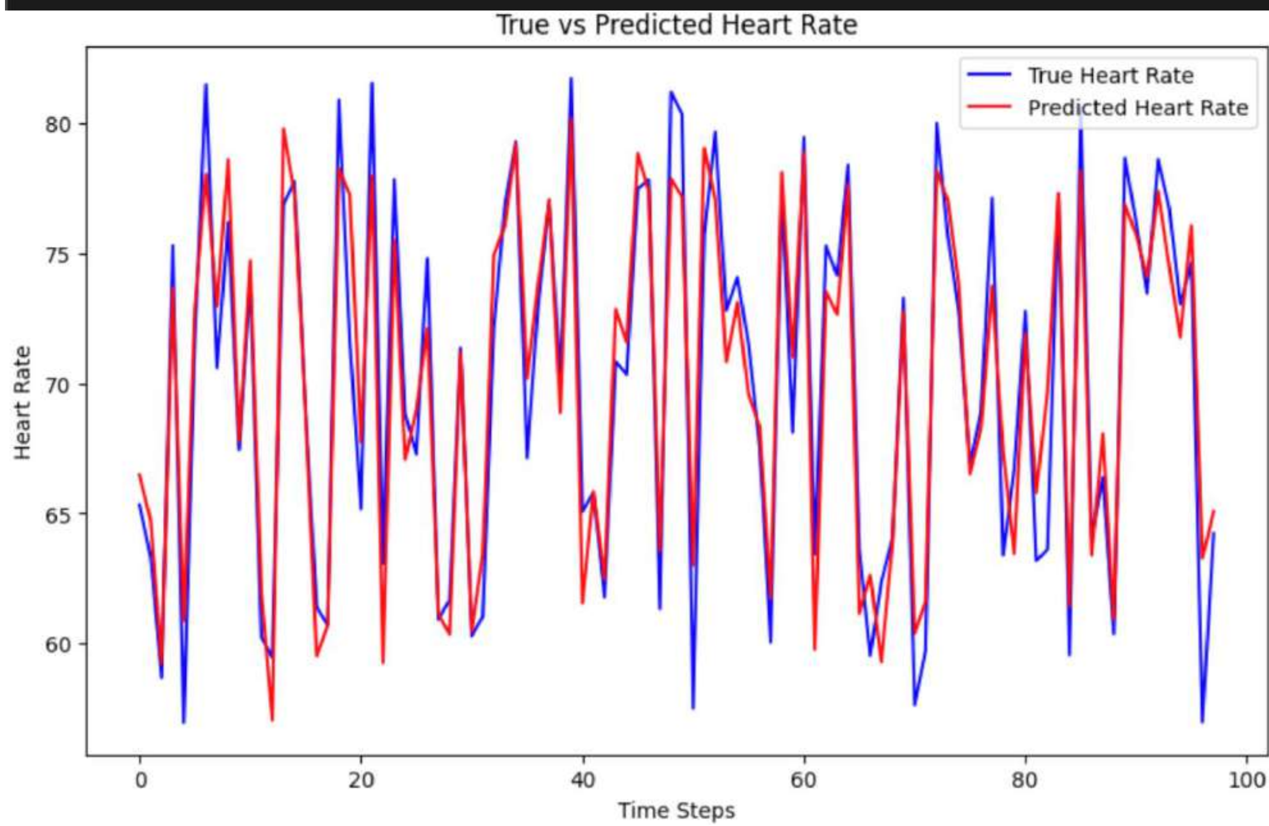
# RNN Architectures: One-to-One



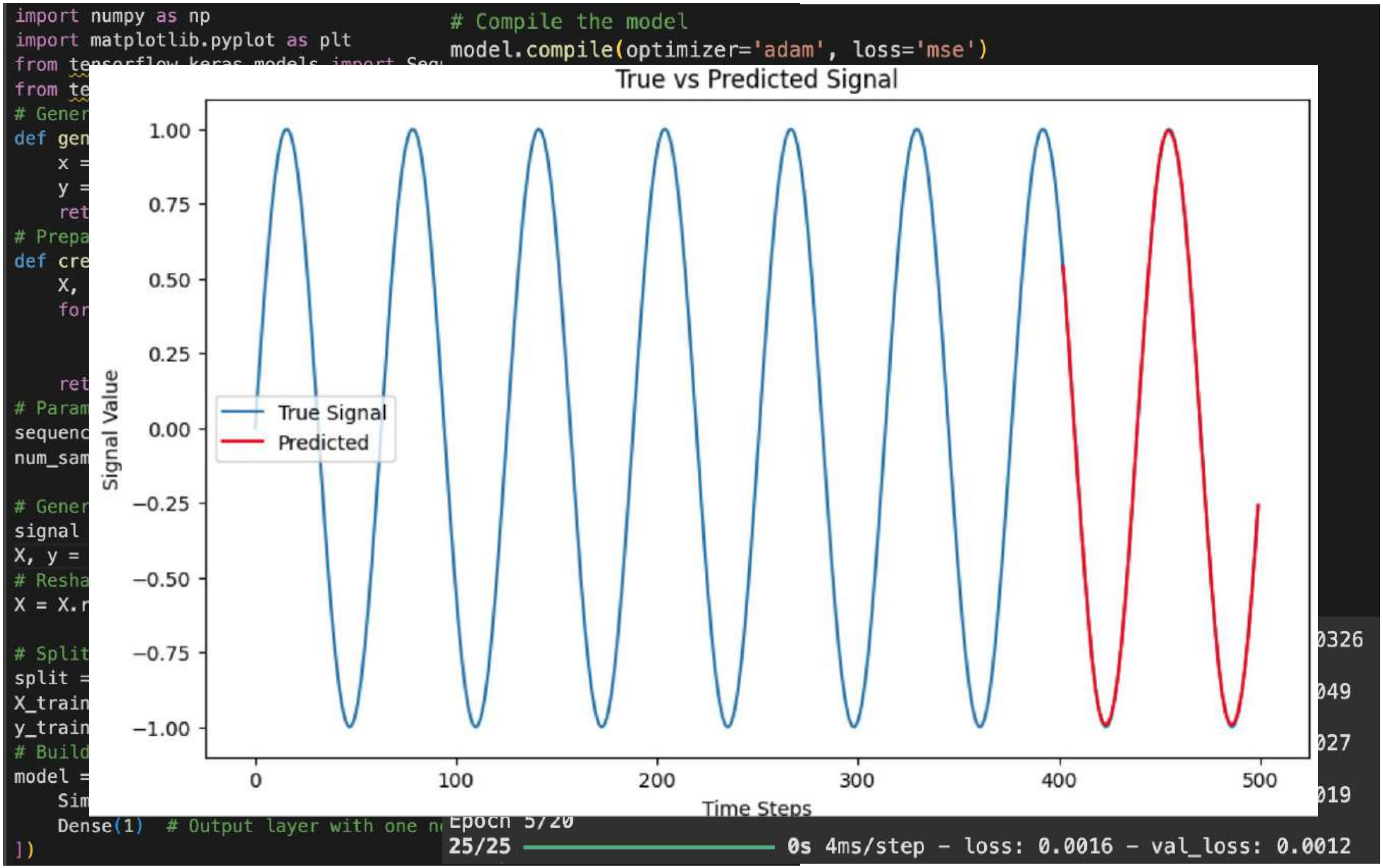

One-to-one:
Traditional NN

without considering temporal dependencies between reviews.

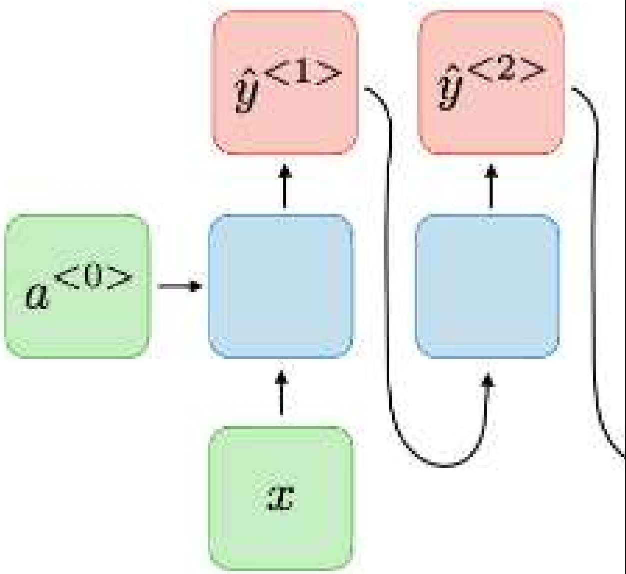
```python
# Build the RNN model
model = Sequential([
    SimpleRNN(50, activation='tanh', input_shape=(sequence_length, 1)),
    Dense(1)  # Output layer with one neuron
])

# Compile the model
model.compile(optimizer='adam', loss='mse')
```



# One-to-One Continued…



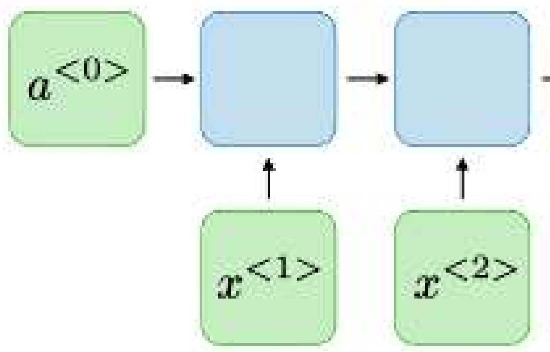
# RNN Architectures: One-to-Many




One-to-Many: Music generation

```python
# Build the model
model = tf.keras.models.Sequential([
    tf.keras.layers.SimpleRNN(128, input_shape=(seq_length, num_chars), return_sequen
    tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(num_chars, activation='soft
])

optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)

# Train the model
model.fit(X, y, batch_size=128, epochs=50)

# Generate text
def generate_text(seed_text, temperature=1.0):
    generated_text = seed_text
    for i in range(400):
        x_pred = np.zeros((1, seq_length, num_chars))
        for t, char in enumerate(seed_text):
            x_pred[0, t, char_to_idx[char]] = 1.0
        preds = model.predict(x_pred, verbose=0)[0][-1]  # Take prediction from the l
        next_index = np.random.choice(len(chars), p=np.exp(np.log(preds) / temperatu
        next_char = idx_to_char[next_index]
        generated_text += next_char
        seed_text = seed_text[1:] + next_char
    return generated_text

# Generate text given an initial line
initial_line = "Tere sang jina yahan, tere sang mar jana"
generated_lyrics = generate_text(initial_line.lower())
print(generated_lyrics)
```

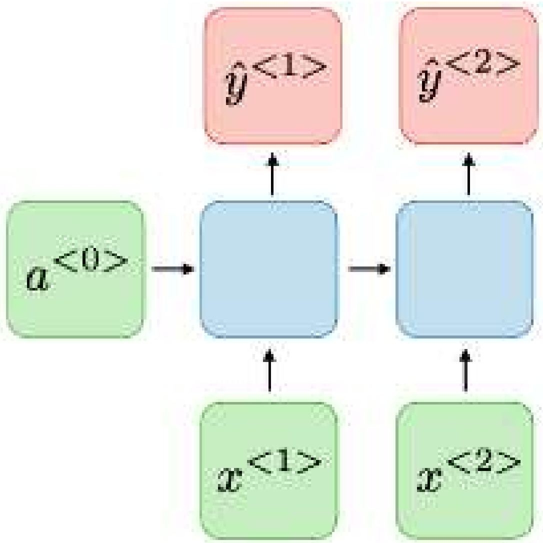
# RNN Architectures: Many-to-one

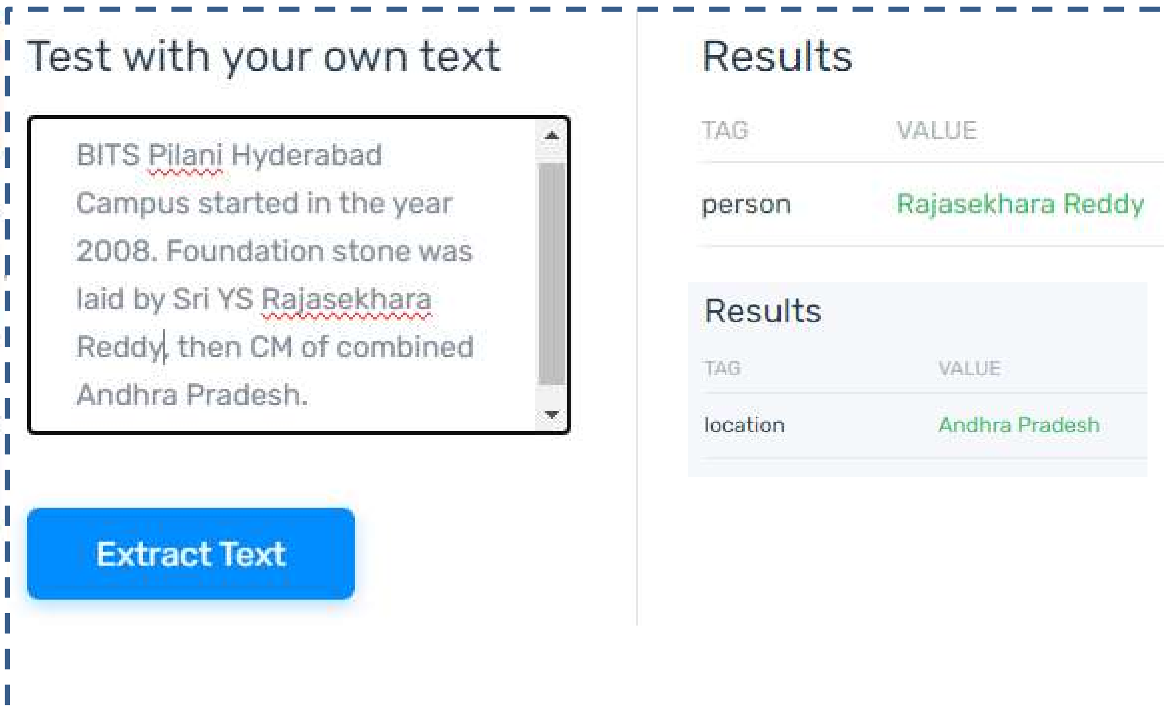



```
# Build the RNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(max_words, 16, input_length=max_sequence_length),
    tf.keras.layers.LSTM(32),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(padded_sequences, labels, epochs=10, batch_size=32)

# Example text for prediction
example_text = "This product is fantastic!"

# Tokenize and pad the example text
example_sequence = tokenizer.texts_to_sequences([example_text])
padded_example_sequence = pad_sequences(example_sequence, maxlen=max_sequence_length,

# Predict sentiment
prediction = model.predict(padded_example_sequence)

# Print prediction
if prediction[0] > 0.5:
    print("Positive Sentiment")
else:
    print("Negative Sentiment")
```

Many-to-One:
Sentiment Classification

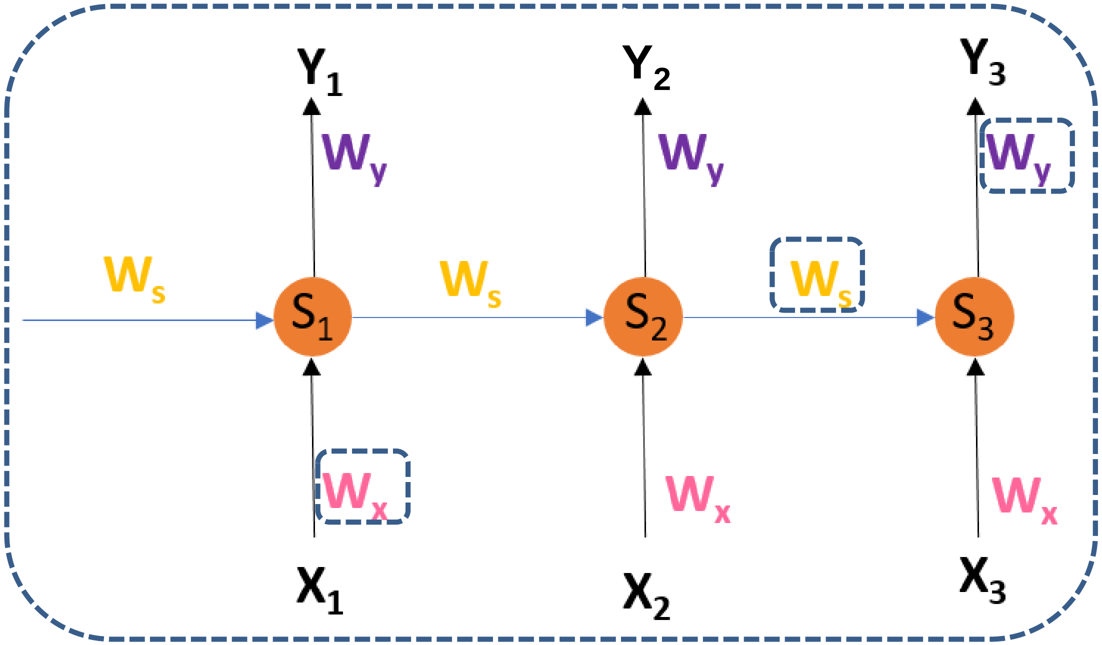
# RNN Architectures: Many-to-Many




Many-to-Many: Named Entity Recognition

# RNN Training: Backpropagation Through Time




$$E = (1/N) \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$$

Adjusting $W_Y$ ($E_3$ is a function of $Y_3$ and $Y_3$ is a function of $W_y$):

$$\frac{\partial E_3}{\partial W_Y} = \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial W_Y}$$

Similarly, now adjusting $W_S$:

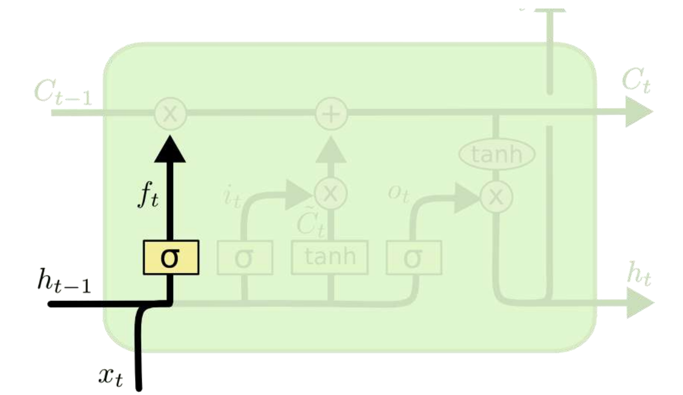
$$\frac{\partial E_3}{\partial W_s} = \underbrace{\left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial W_s} \right\}}_{E_3 \text{ --- } Y_3 \text{ --- } S_3 \text{ --- } W_s} + \underbrace{\left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial S_2} \cdot \frac{\partial S_2}{\partial W_s} \right\}}_{\text{With respect to } S_2} + \underbrace{\left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial S_2} \cdot \frac{\partial S_2}{\partial S_1} \cdot \frac{\partial S_1}{\partial W_s} \right\}}_{\text{With respect to } S_1}$$

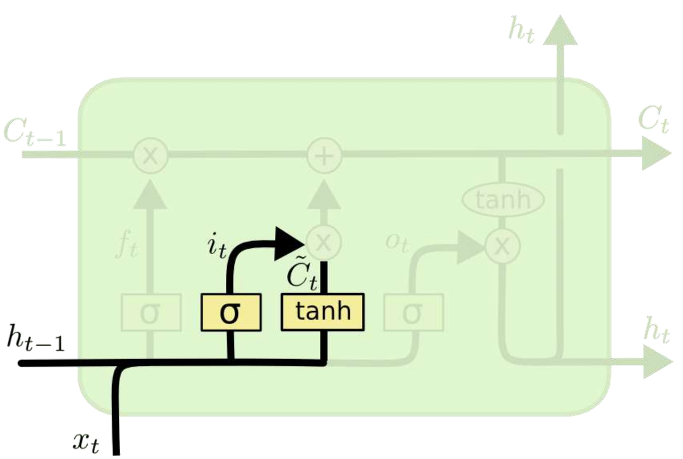
For $W_x$:

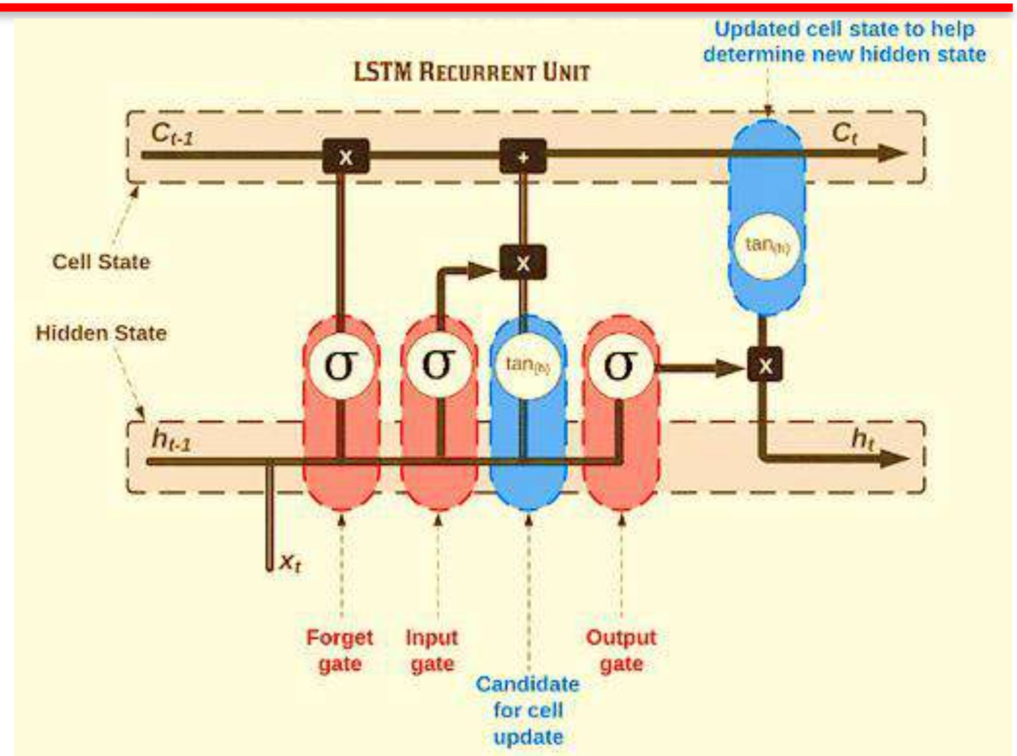
$$\frac{\partial E_3}{\partial W_x} = \left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial W_x} \right\} + \left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial S_2} \cdot \frac{\partial S_2}{\partial W_x} \right\} + \left\{ \frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial S_2} \cdot \frac{\partial S_2}{\partial S_1} \cdot \frac{\partial S_1}{\partial W_x} \right\}$$

# Long Short Term Memory: Ex. RNN

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

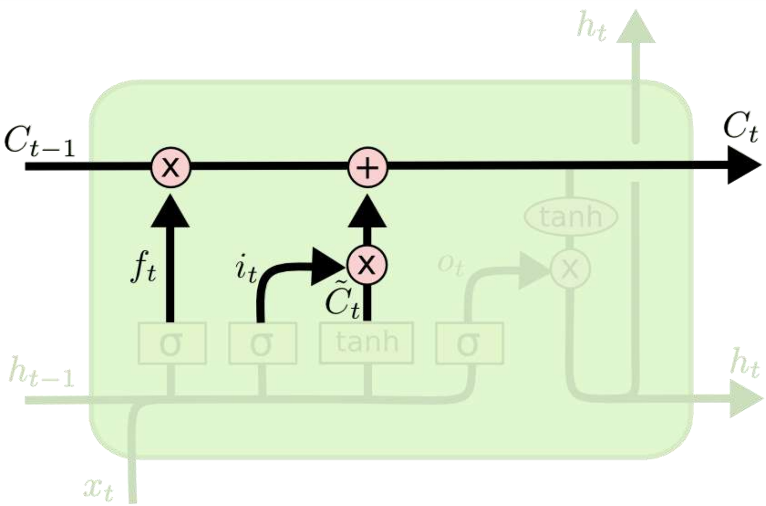
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Img. Source: https://www.simplilearn.com/

- Problems with RNNs: Vanishing and Exploding gradients. LSTMs solve vanishing prob. Exploding gradients prob. may be solved by Gradient clipping or regularization etc.

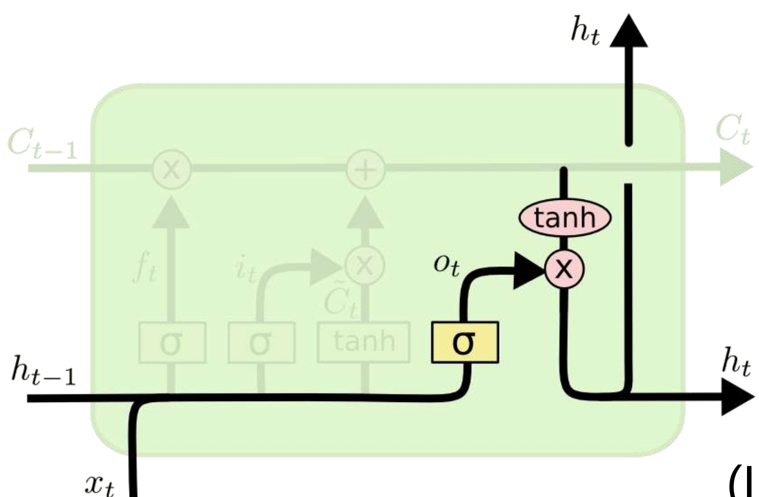
# Continued...

Assignment 5: 23rd Nov 2024




Update $C_{t-1}$ into $C_t$.

$$C_t = \underbrace{\boxed{f_t * C_{t-1}}}_{\text{Forgetting the things}} + \underbrace{i_t * \tilde{C}_t}_{\text{New value}}$$




Output information relevant to a verb

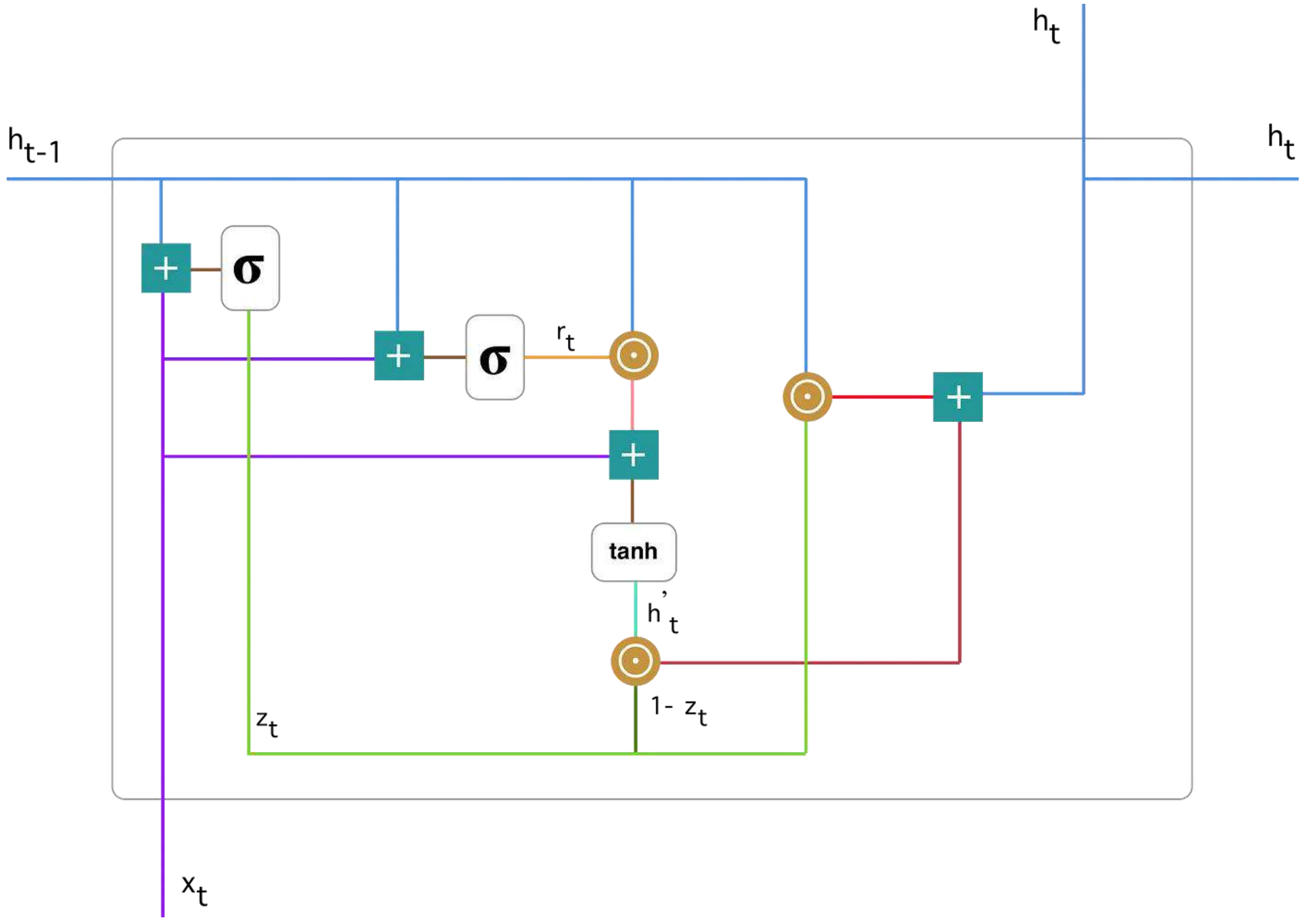
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

(I loved the food. But, the service was terrible)

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

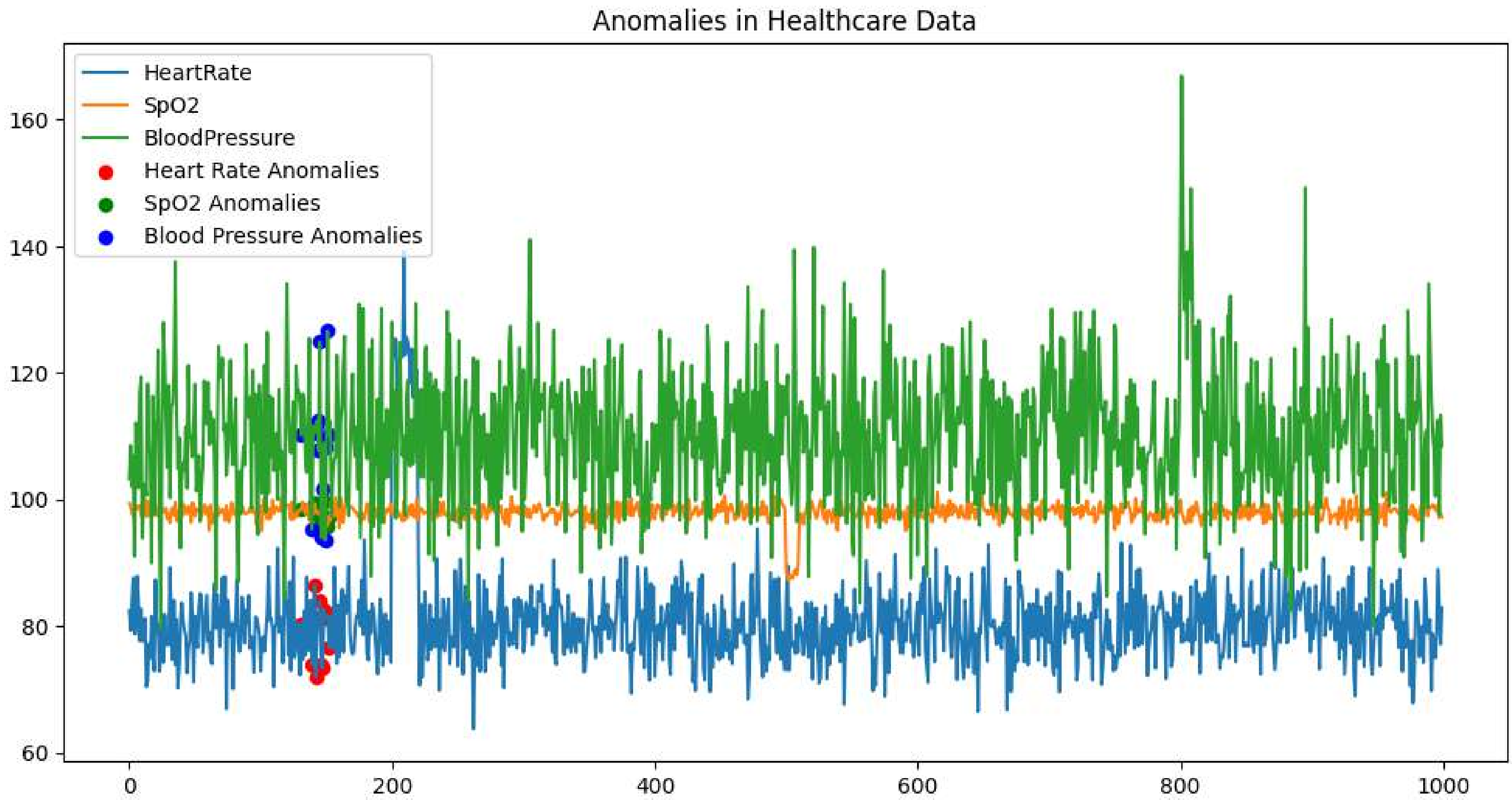
# Gated Recurrent Unit (GRU)




$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

# Anomaly Detection

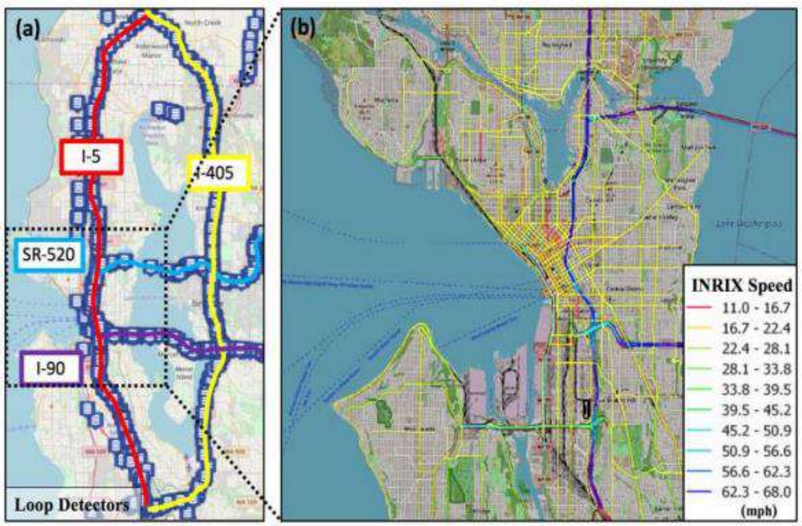
```
# 3. Build the LSTM Autoencoder Model
model = Sequential([
    # Encoder
    LSTM(64, return_sequences=True, input_shape=(seq_length, X.shape[2])),
    Dropout(0.2),
    LSTM(32, return_sequences=False),

    # Decoder
    RepeatVector(seq_length),
    LSTM(64, return_sequences=True),
    Dropout(0.2),
    TimeDistributed(Dense(X.shape[2]))
])
```

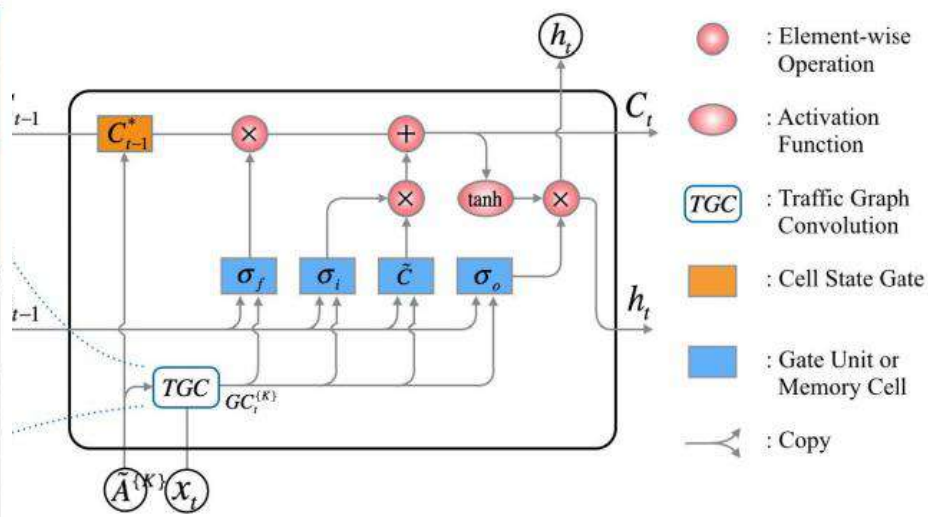


# Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting

Zhiyong Cui, *Student Member, IEEE*, Kristian Henrickson, Ruimin Ke, *Student Member, IEEE*, and Yinhai Wang, *Senior Member, IEEE*

ConvLSTM: Network wide traffic states are identified with most influential roadways.




FFR: Free Flow Reachability (i.e vehicle speed) info, $\tilde{A}$: neighbourhood information, W: weights.

sensors 2022

Article

# A Correlation-Based Anomaly Detection Model for Wireless Body Area Networks Using Convolutional Long Short-Term Memory Neural Network

Albatul Albattah [1] and Murad A. Rassam [1,2,*]

$$f_t = \sigma(W_{xf} \otimes X_t + W_{hf} \otimes H_{t-1} + W_{cf} \odot C_{t-1} + B_f) \tag{8}$$

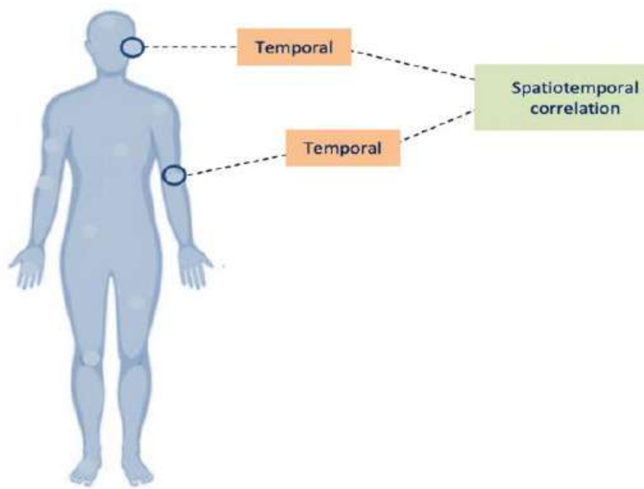
$$i_t = \sigma(W_{xi} \otimes X_t + W_{hi} \otimes H_{t-1} + W_{ci} \odot C_{t-1} + B_i) \tag{9}$$

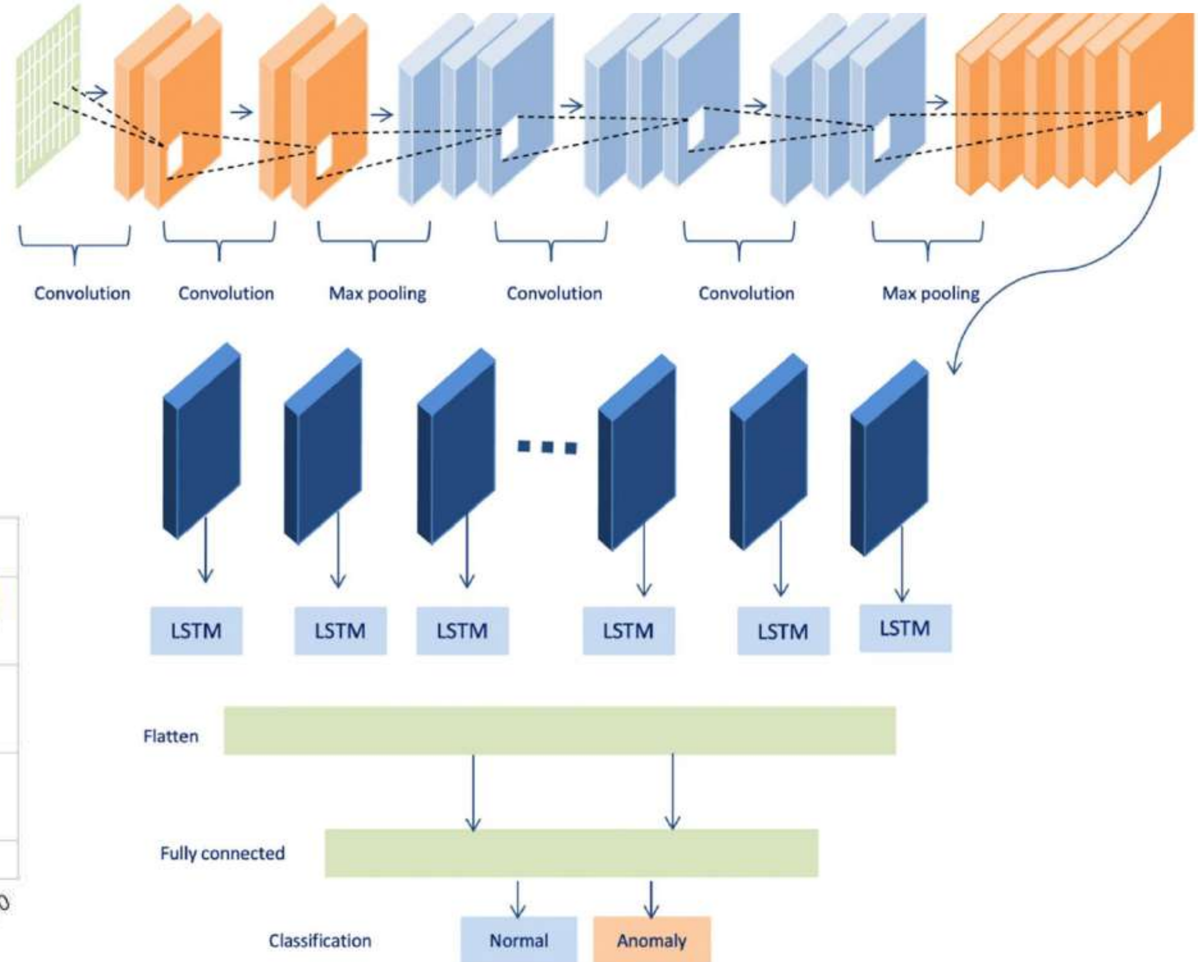
$$c'_t = \tanh(W_{xc} \otimes X_t + W_{hc} \otimes H_{t-1} + B_c) \tag{10}$$

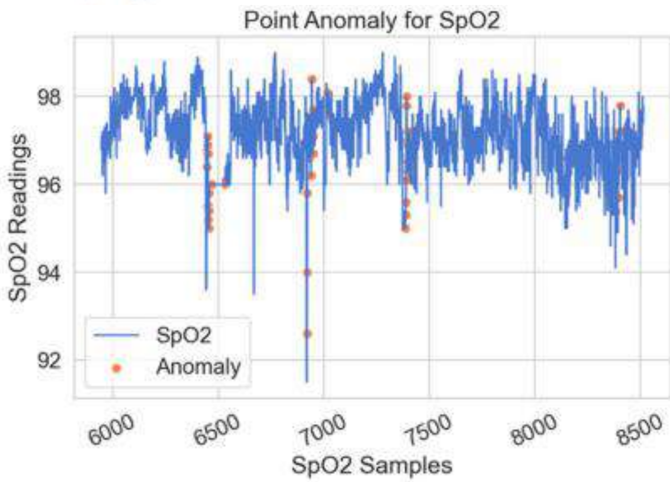
$$c_t = f_t \odot C_{t-1} + c'_t \tag{11}$$

$$o_t = \sigma(W_{xo} \otimes X_t + W_{ho} \otimes H_{t-1} + W_{co} \odot C_t + B_o) \tag{12}$$

$$h_t = o_t \odot \tanh(c_t) \tag{13}$$




Convolution + LSTM: genre of a movie by seeing the trailer (Horror or Detective)

# Autoregressive Models

- A generative model that generates new data points by regressing each observation on previous observations within the series.

- Mathematically: $\boxed{Y_t} = \Phi_0 + \Phi_1.Y_{t-1} + \xi_t$

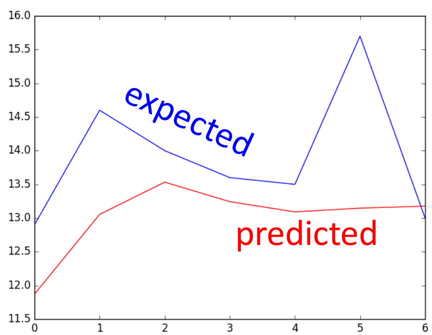
Current value

Previous value

Where, $\Phi_0$, $\Phi_1$ are coefficients to be estimated. $\xi_t$ is error term at time 't'.

AR(1): Order 1

What would be AR(2), AR(3),…



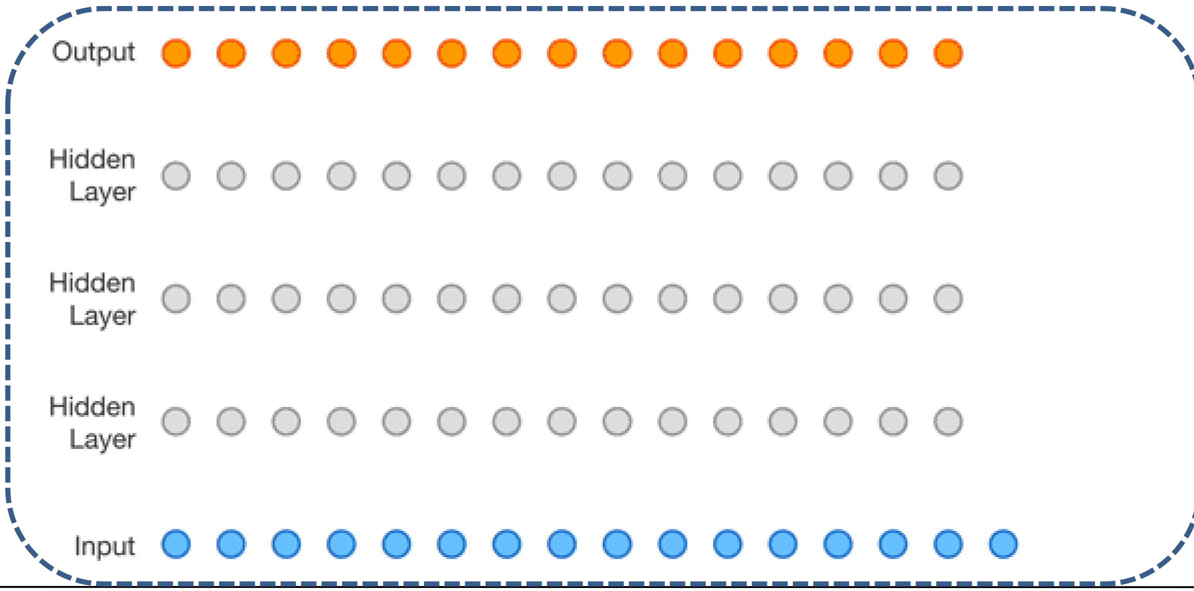




Image source: WaveNet (Google DeepMind)

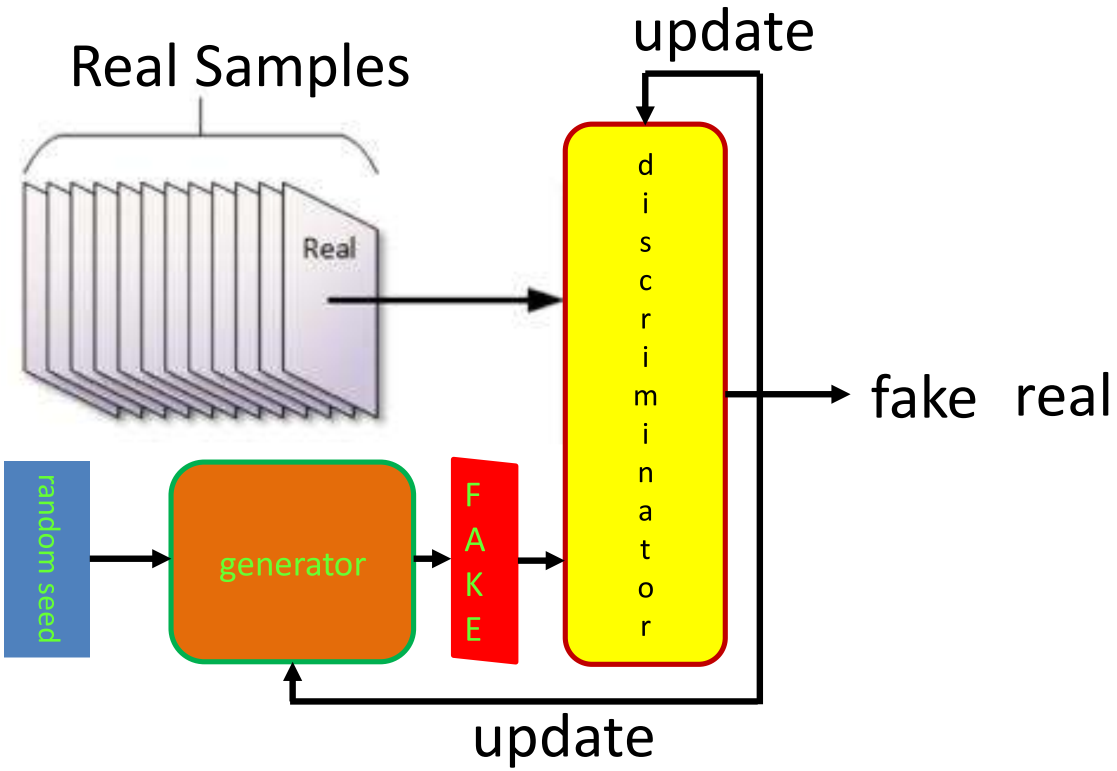
Probabilistically, Autoregression can be expressed as: $p(x) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots x_{i-1}) = \prod_{i=1}^{n} p(x_i|x_{<i})$

# Auto-regressive (Statistical) Vs LSTM (DL)

| Scenario | AR Model | LSTM |
|---|---|---|
| Linear relationships | ✅ Ideal | ⚠️ Overkill |
| Stationary data | ✅ Ideal | ✅ Works well |
| Non-linear relationships | ⚠️ Limited capability | ✅ Ideal |
| Long-term dependencies | ⚠️ Poor performance | ✅ Excellent |
| Small datasets | ✅ Suitable | ⚠️ Risk of overfitting |
| Large, complex datasets | ⚠️ Limited capability | ✅ Ideal |

# Generative Adversarial Network (GAN)

- GANs are neural networks (Deep ANNs) that learn to create synthetic data similar to some known input data. Or have the capability to generate new data.
- Ex: Admission office (Discriminator) checking the transcripts of newly admitted students( Generator)…



# Loss Functions in a GAN

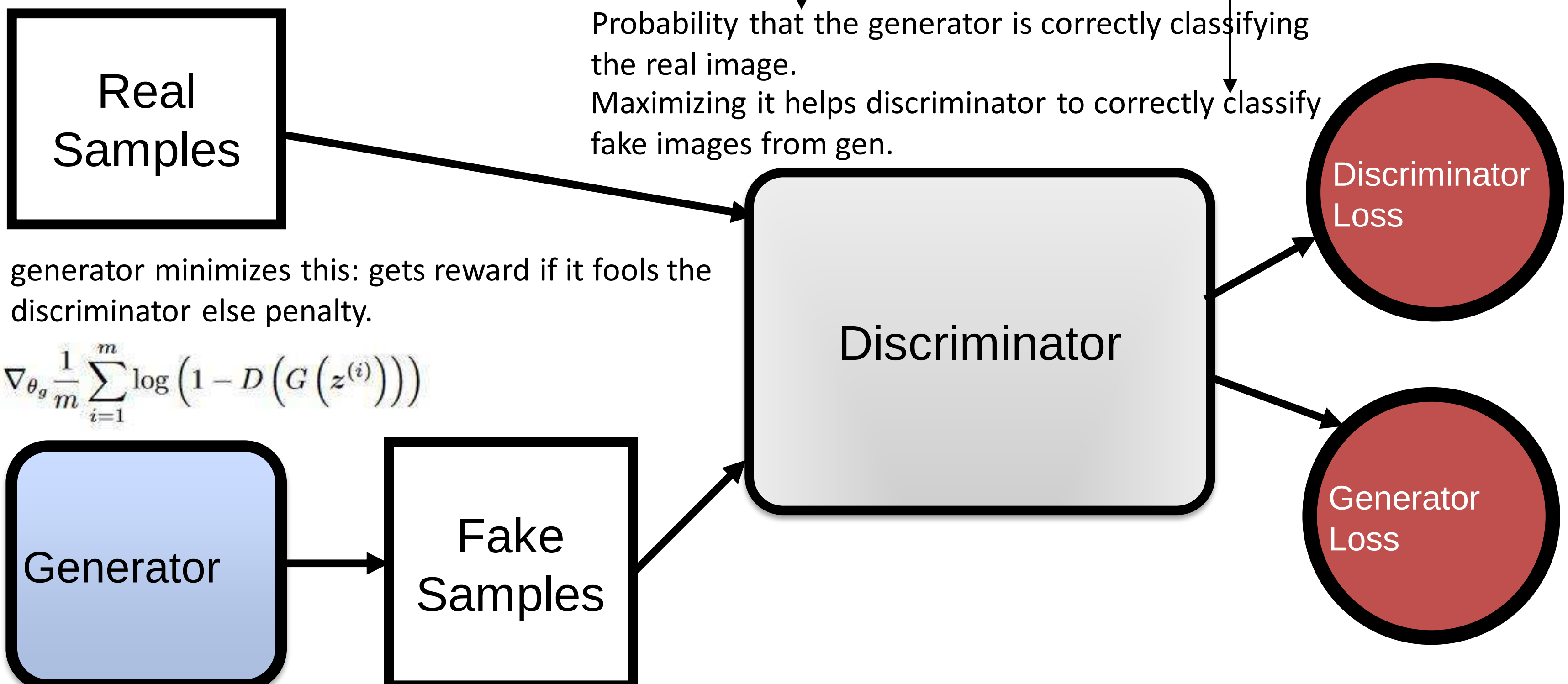
$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

Probability that the generator is correctly classifying the real image.

Maximizing it helps discriminator to correctly classify fake images from gen.

generator minimizes this: gets reward if it fools the discriminator else penalty.

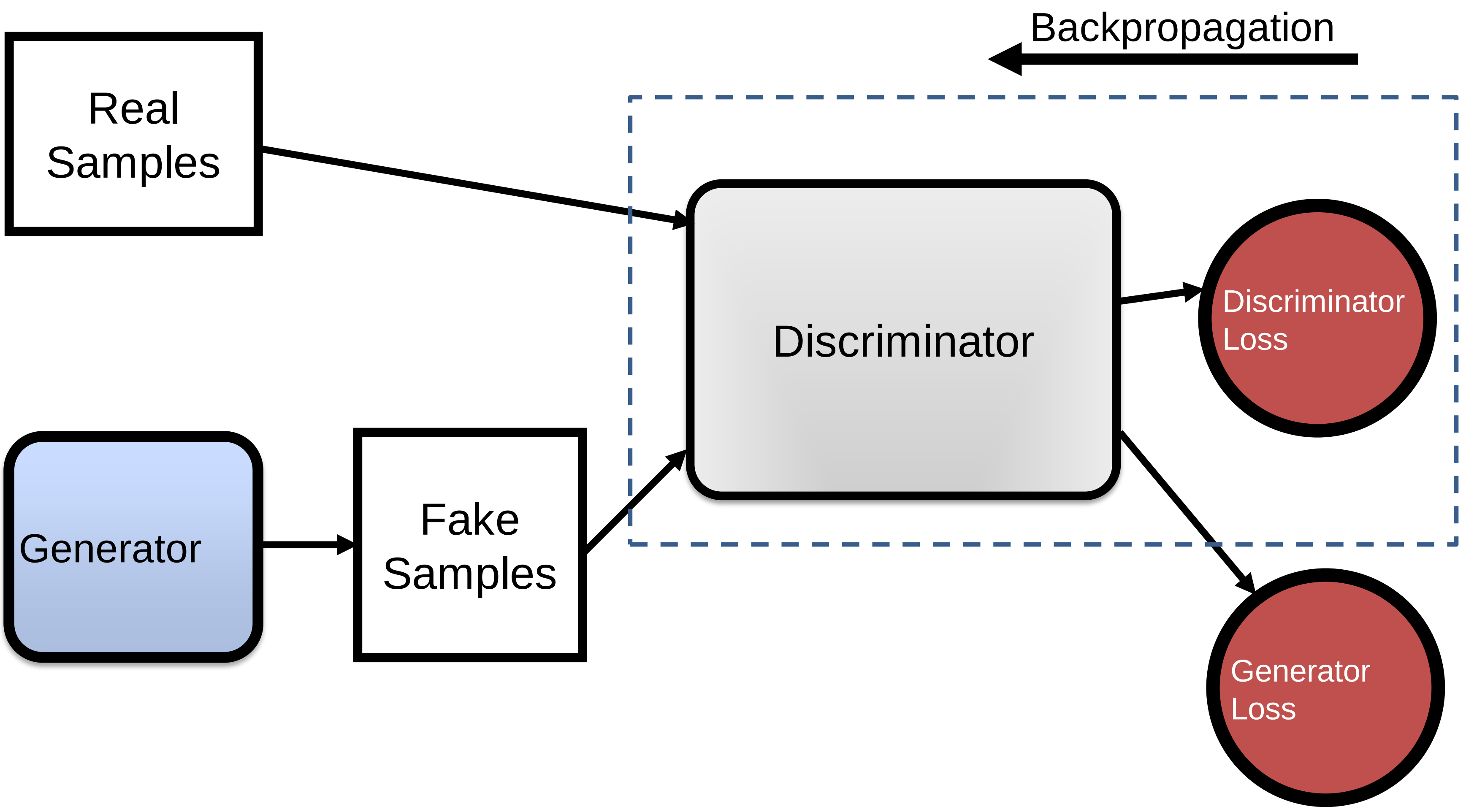
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$



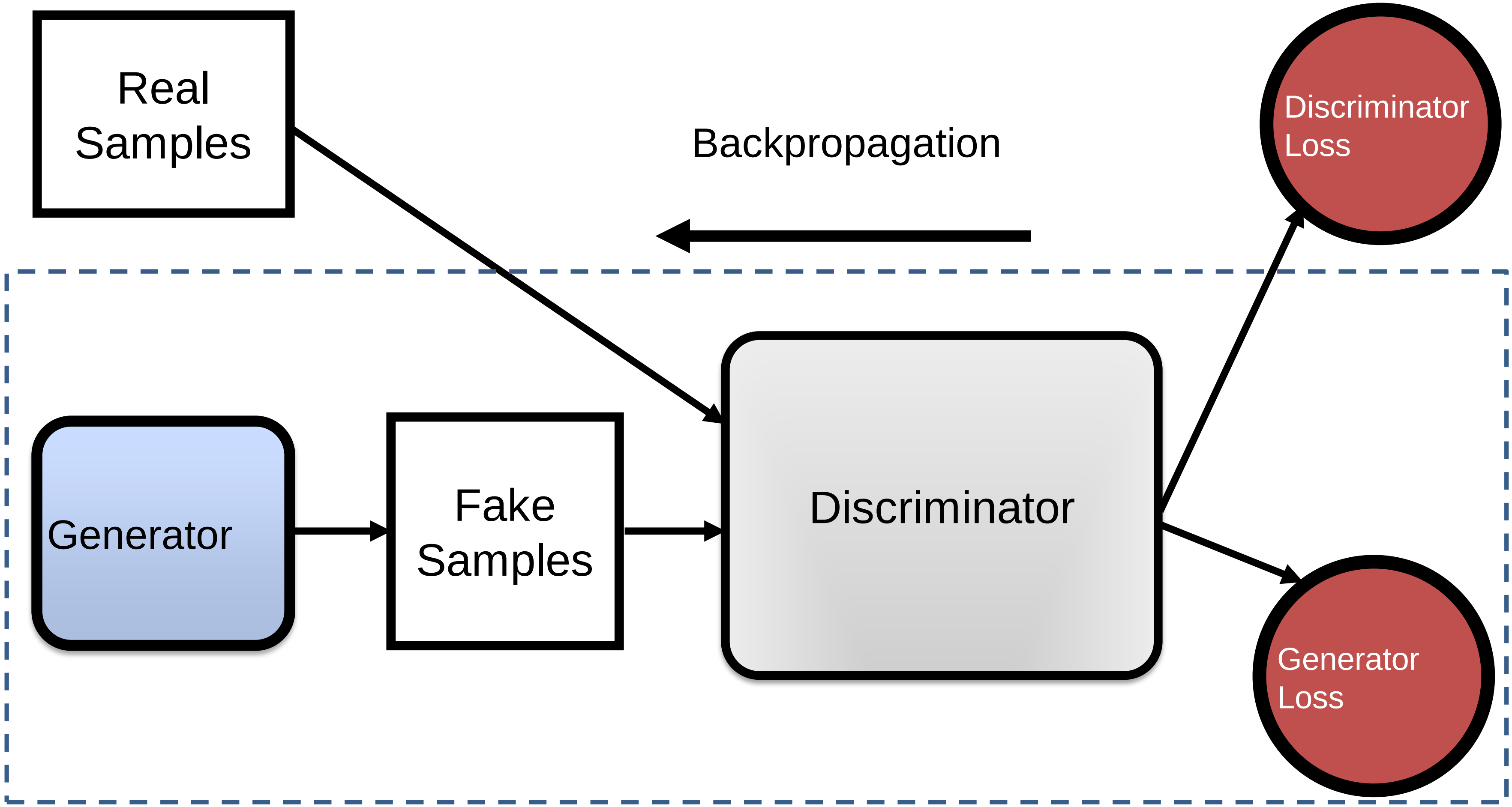

$E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$

D(x): Discriminator's prob. that x is real; $E_x$: Expected value over all real x; G(z): Generator's output when given noise is z; D(G(z)): Discriminator's prob. that a fake instance is real; $E_z$: Expected value over all random inputs to the generator.

# Backpropagation in GAN

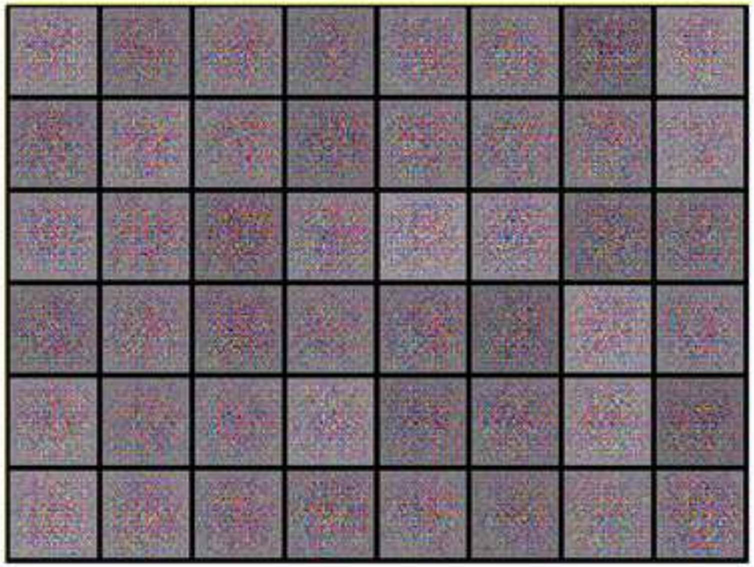


# Backpropagation in GAN

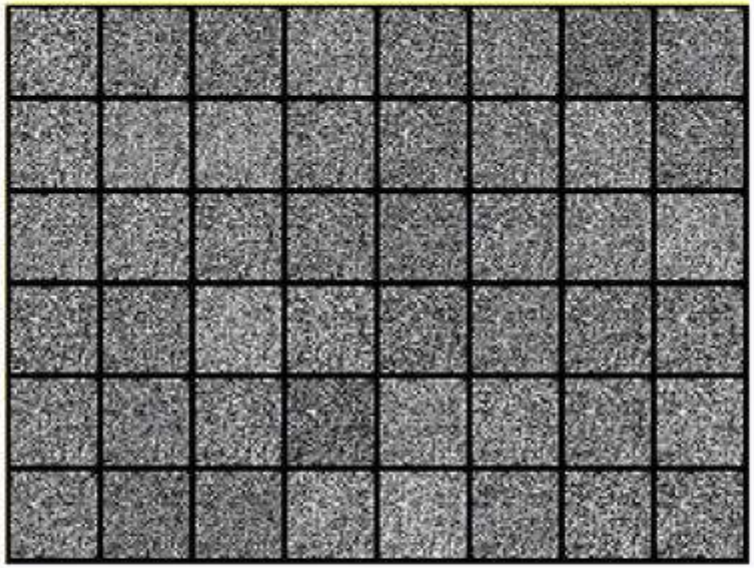


# GAN Applications

**Applications:** synthetic image/ video generation (deepfake), Image-to-image translation, Anomaly detection,...



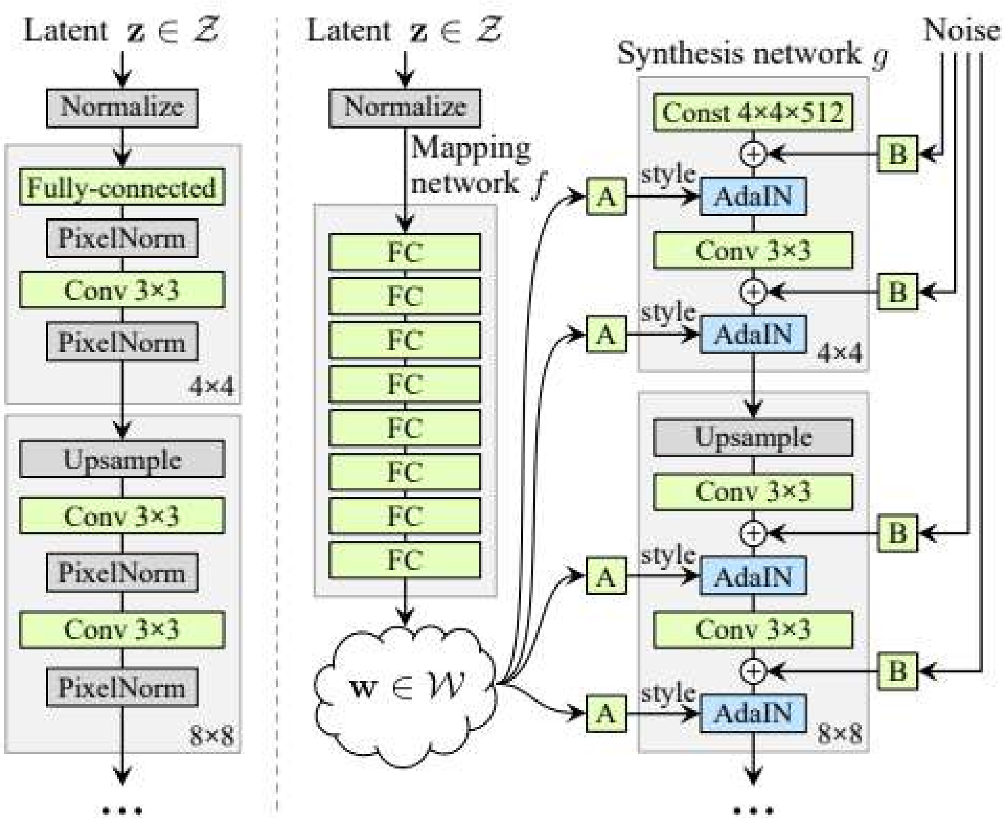
(Deep Convolutional GAN: Better Quality)



(Generated Vanilla GAN images in MNIST)

Image Source: https://theaisummer.com

# StyleGAN: NVIDIA




(a) Traditional

(b) Style-based generator

https://github.com/NVlabs/stylegan

Thank You!