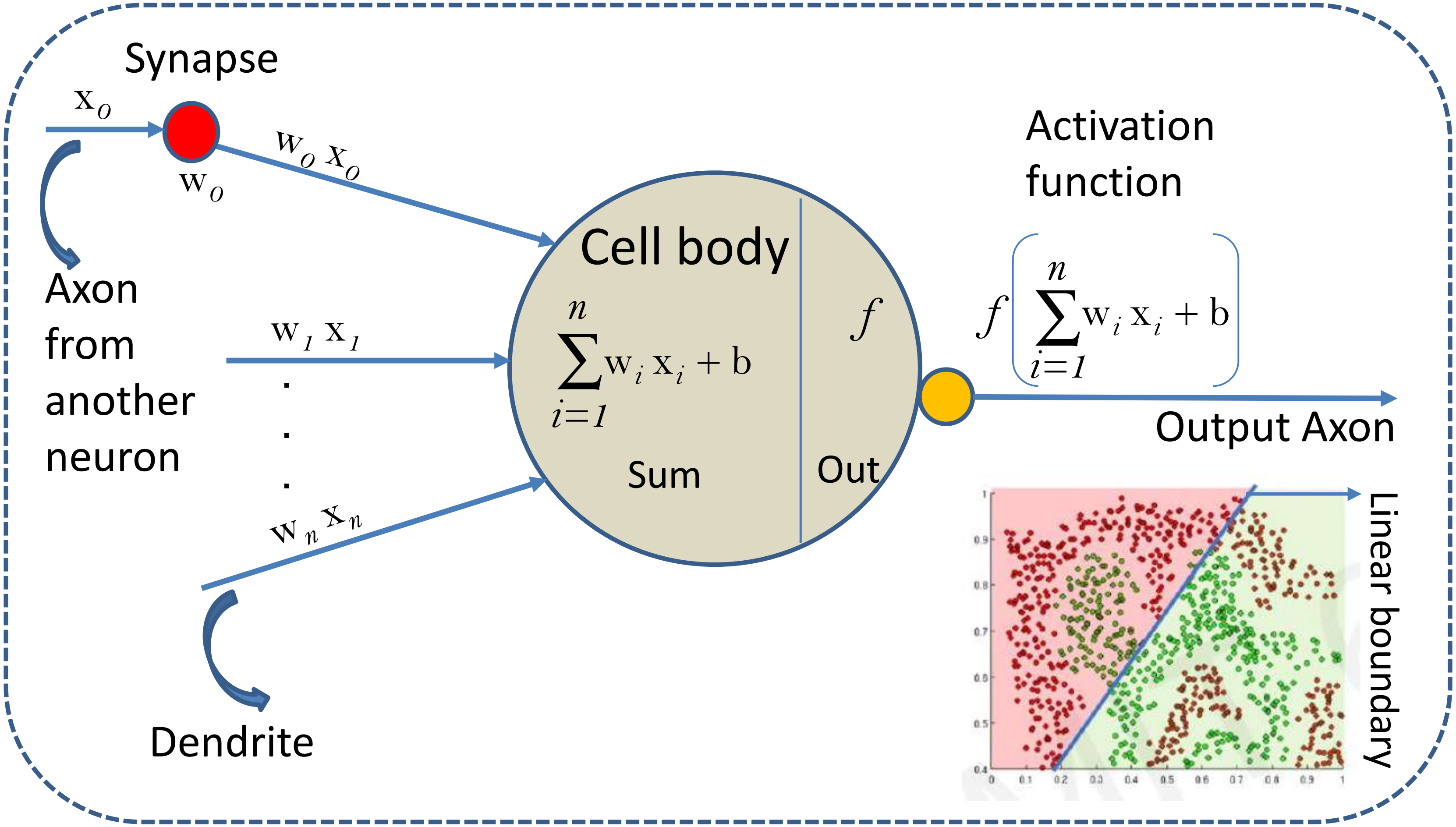
16.04.2024

BITS F464: Machine Learning

NEURAL NETWORKS: CONVOLUTIONAL/ RECURRENT/ GENERATIVE MODELS

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

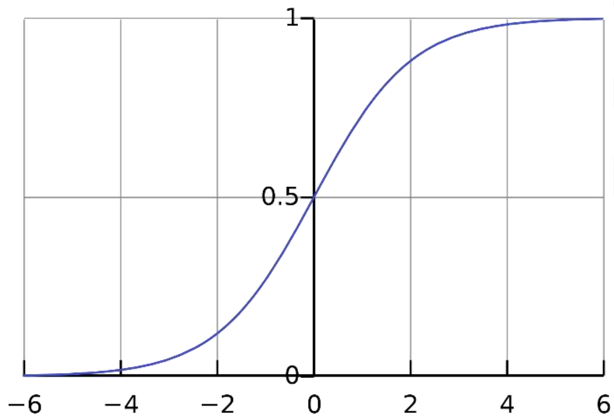
Recap: A Perceptron



Modelling Mathematically a Neuron

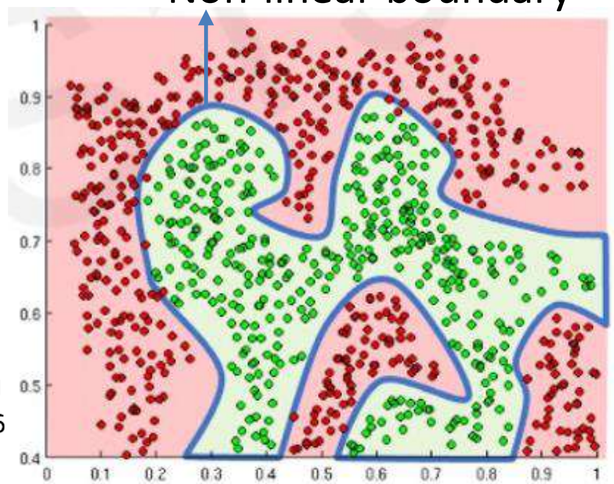
Vanishing Gradient in MLPs/BPNs

$\sigma(z) = 1/1+e^{-z}$



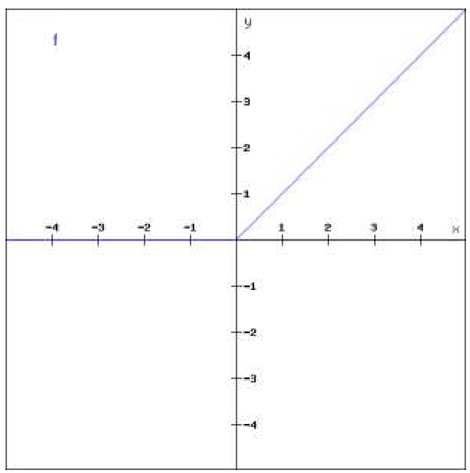
Sigmoid function/ Logistic Curve

Non-linear boundary



Approx. Arbitrary Complex fun

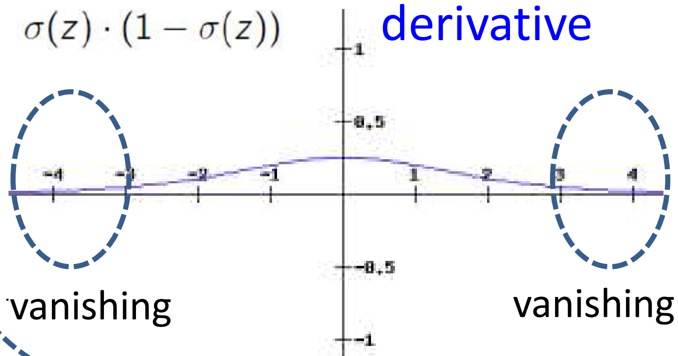
$\mathrm{ReLU}(z) = \max(0, z)$



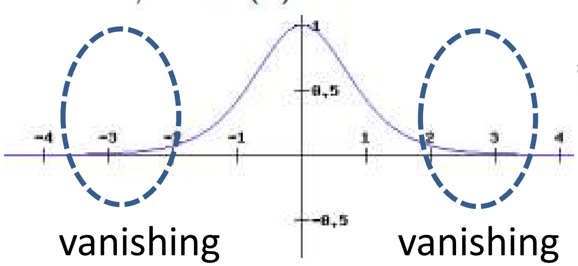
Leaky ReLU(z) = 0.01z, z < 0
= z, z >= 0

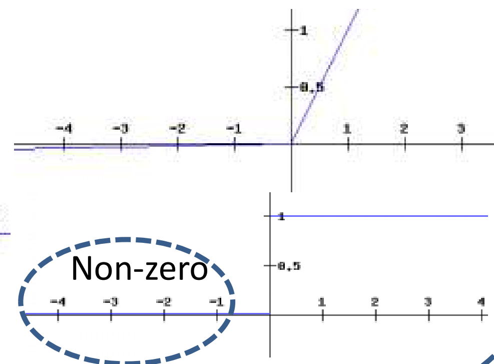
$\sigma(z) \cdot (1 - \sigma(z))$

derivative

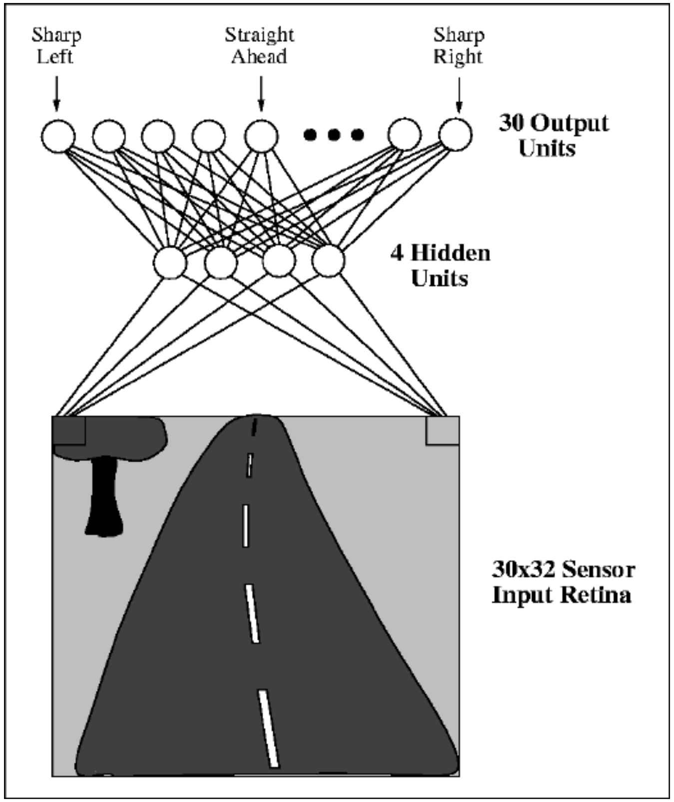


$1/\cosh^2(z)$





ALVINN: An Autonomous Land Vehicle In a Neural Network



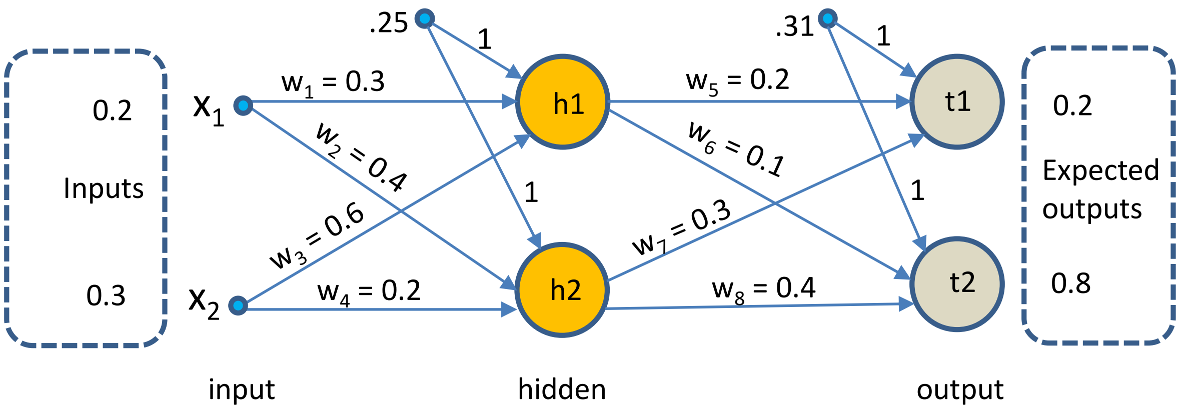
Source: https://www.ri.cmu.edu/ (1989: 3-layer Network)

An application of a Backpropagation Neural Network in smart driving

Example Backpropagation Neural Networks

Quiz Question

Learning rate: $\eta = 0.4$



Total error:

$E_{total} = ½\sum(\text{target-output})^2$

$E1 = ½(0.2 - .6486)2 = .1006$

$E_2 = ½(0.8 - .6480)^2 = .0116$

$E_{total} = .1006 + .0116 = 0.1122$

Forward Pass:

For h1:

Sum = .25x1 + .3x.2 + .3x.6 = 0.49

Output = $1/1+e^{-.49}$ = 0.6201

For h2:

Sum = .25x1 + .2x.4 + .3x.2 = 0.39

Output = $1/1+e^{-.39}$ = 0.5963

For t1:

Sum = .31x1 + .6201x.2 + .5963x.3 = .6129

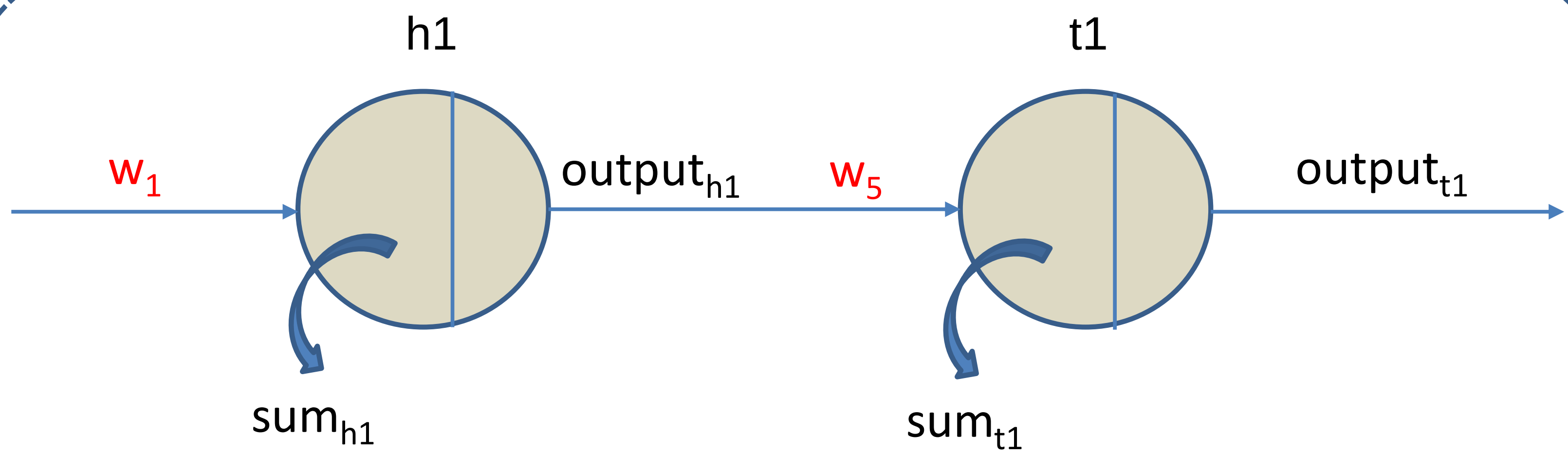
Output = $1/1+e^{-.6129}$ = 0.6486

For t2:

Sum = .31x1 + .6201x.1 + .5963x.4 = .6105

Output = $1/1+e^{-.6105}$ = 0.6480

Example Chain Rule



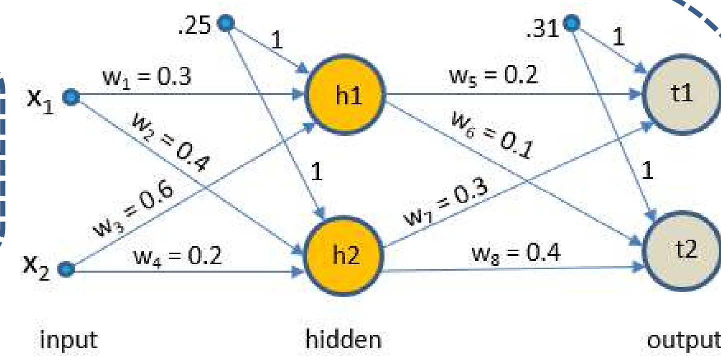
Chain Rule:

E is affected by → $output_{t1}$ → sum_{t1} → $output_{h1}$ → sum_{h1} → w_1

Backward Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial output_{t1}} \times \frac{\partial output_{t1}}{\partial sum_{t1}} \times \frac{\partial sum_{t1}}{\partial w_5}$$

... Eq. (1)



$$E_{total} = \tfrac{1}{2} \sum (\text{actual} - \text{observed})^2$$

$$= \tfrac{1}{2}\left\{(actual_{t1} - output_{t1})^2 + \underbrace{(actual_{t2} - output_{t2})^2}\right\}$$

$$\frac{\partial output_{t1}}{\partial sum_{t1}} = ?$$

$$\sigma(x) = 1/1+e^{-x}$$

$$d/dx(\sigma(x)) = \sigma(x)\ (1- \sigma(x))$$

$$\frac{\partial E_{total}}{\partial output_{t1}} = 2 \times \tfrac{1}{2}\ (actual_{t1} - output_{t1}) \times -1 + 0$$

$$= (output_{t1} - actual_{t1})$$

$$= 0.6486 - 0.2 = 0.4486$$

... Eq. (2)

$$\frac{\partial output_{t1}}{\partial sum_{t1}} = output_{t1}(1\text{-}output_{t1})$$

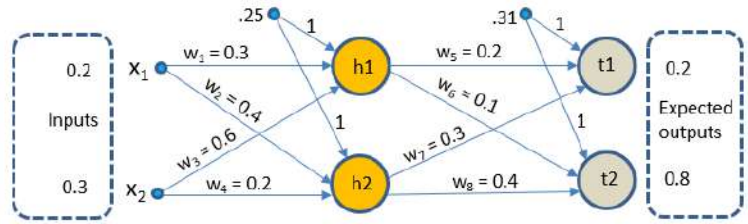
$$=.6486(1-.6486)=.2279$$

... Eq. (3)

$$\frac{\partial sum_{t1}}{\partial w_5} = \frac{\partial (output_{h1} x w_5 + output_{h2} x w_7)}{\partial w_5} = output_{h1} = 0.6201$$

... Eq. (4)

Continued…



Eq. (1): $\frac{\partial E_{total}}{\partial w_5}$ = Eq.(1) x Eq.(2) x Eq.(3) = .4486 x .2279 x .6201 = .0634

$$w_5 = w_5 - \eta \frac{\partial E_{total}}{\partial w_5} = .2 - .4 \times .0634 = \boxed{.1746}$$

$$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial output_{t2}} \text{ X } \frac{\partial output_{t2}}{\partial sum_{t2}} \text{ X } \frac{\partial sum_{t2}}{\partial w_6} = (.6480-.8)x(.6480x(1-.6480))x.6201$$

$$=-.152x.2281x.6201 = -.0215$$

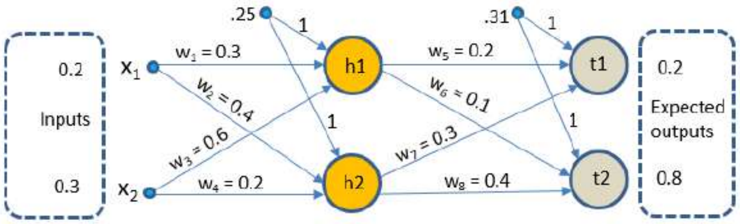
$$w_6 = w_6 - \eta \frac{\partial E_{total}}{\partial w_6} =.1-(.4x-.0215)=\boxed{.1086}$$

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial output_{t1}} \text{ X } \frac{\partial output_{t1}}{\partial sum_{t1}} \text{ X } \frac{\partial sum_{t1}}{\partial w_7} = \text{Eq.(2) x Eq.(3) x } output_{h2}$$

$$= 0.4486 \times 0.2279 \times 0.5963 = 0.0609$$

$$w_7 = w_7 - \eta \frac{\partial E_{total}}{\partial w_7} = .3 - .4 \times .0609 = 0.3 - 0.02436 = \boxed{0.2756}$$

Continued…



.5963

$$\frac{\partial E_{total}}{\partial w_8} = \frac{\partial E_{total}}{\partial output_{t2}} \times \frac{\partial output_{t2}}{\partial sum_{t2}} \times \frac{\partial sum_{t2}}{\partial w_8} = (-.152) x\ .2281 x output_{h2} = -0.0207$$

$$w_8 = w_8 - \eta \frac{\partial E_{total}}{\partial w_8} = 0.4 + 0.4 \times 0.0207 = 0.4 + 0.0083 = \boxed{.4083}$$

Similarly, w_2=.4007 … for you

Now Compute Weights in the Hidden Layer (w_1, w_2, w_3, and w_4): Chain becomes longer.

For w_1: $$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_1}{\partial w_1} + \frac{\partial E_2}{\partial w_1}$$

$$w_1 = w_1 + \eta\left(\partial E_{total} / \partial w_1\right)$$
$$= 0.3 - 0.4 \times 0.0008 = \boxed{.2997}$$

Where, $$\frac{\partial E_1}{\partial w_1} = \frac{\partial E_1}{\partial output_{t1}} \times \frac{\partial output_{t1}}{\partial sum_{t1}} \times \frac{\partial sum_{t1}}{\partial output_{h1}} \times \frac{\partial output_{h1}}{\partial sum_{h1}} \times \frac{\partial sum_{h1}}{\partial w_1}$$

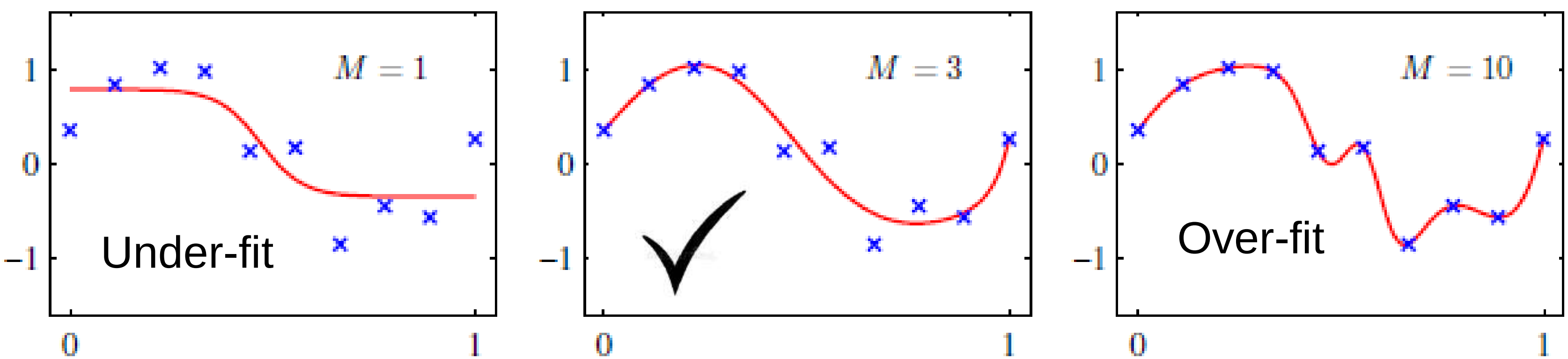
$$\frac{\partial E_1}{\partial w_1} = .4486 \times .2279 \times w5 \times (output_{h1} \times (1 - output_{h1})) \times 0.2 = 0.00096$$

Now, $$\frac{\partial E_2}{\partial w_1} = \frac{\partial E_2}{\partial output_{t2}} \times \frac{\partial output_{t2}}{\partial sum_{t2}} \times \frac{\partial sum_{t2}}{\partial output_{h1}} \times \frac{\partial output_{h1}}{\partial sum_{h1}} \times \frac{\partial sum_{h1}}{\partial w_1}$$

$= -.1520 \times .2281 \times w6 \times .2356 \times .2 = -.00016$ $\partial E_{total}/\partial w_1$ =.00096 - .00016=.0008

Regularization in Neural Networks

- Which one is a free parameter in a Neural network?
  - Input Output or Number of units in the hidden layer (M)
- Why Regularization is needed in Neural Networks?
  - To improve the generalization/ learning outcome. To control impact of noise and fluctuations on the dataset. Alternatively, to avoid over-fitting.

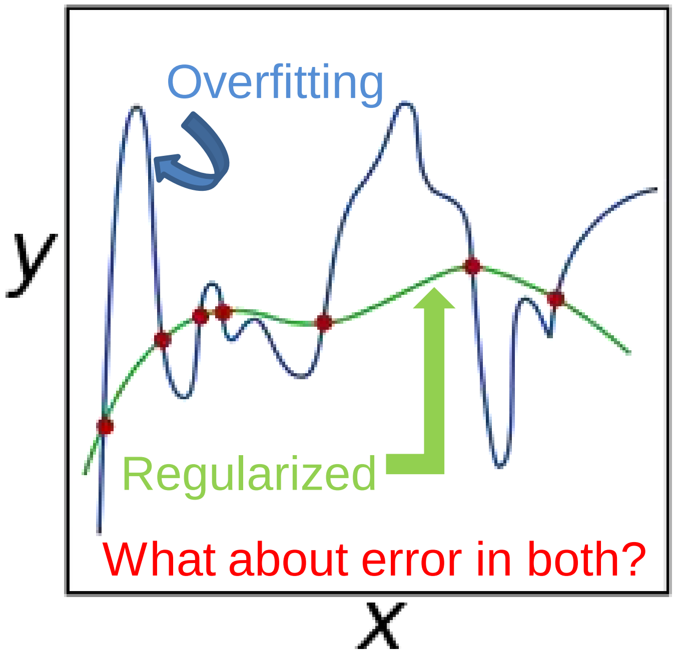


(Fitting a Sinusoidal dataset with different number of hidden units and Sum-of-Squares error function optimized by Gradient descent)

Img. Source: Bishop Text

Regularization: Weight Decay (L1/L2)

- Control model complexity by the addition of a regularization term to the error function.



Data term Regularization term

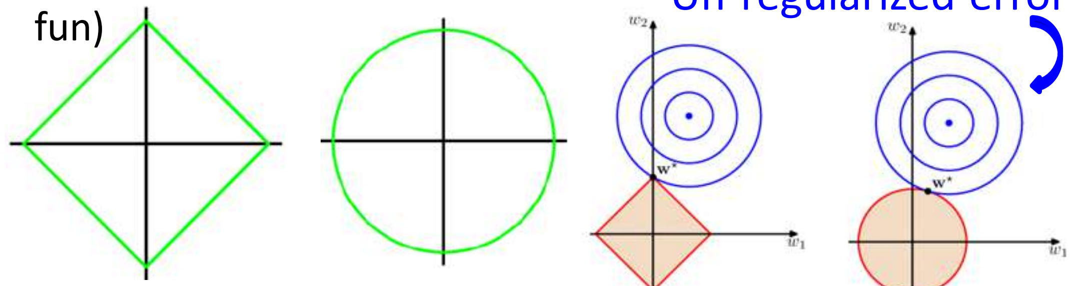
Loss function: $\boxed{E_D(\mathbf{w})} + \boxed{\lambda E_W(\mathbf{w})}$

With Sum-of-squares error and a quadratic regularizer: $\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \boxed{\frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}}$

Contour plots (behaviour of fun)

L2 regularization (Ridge)

Un-regularized error

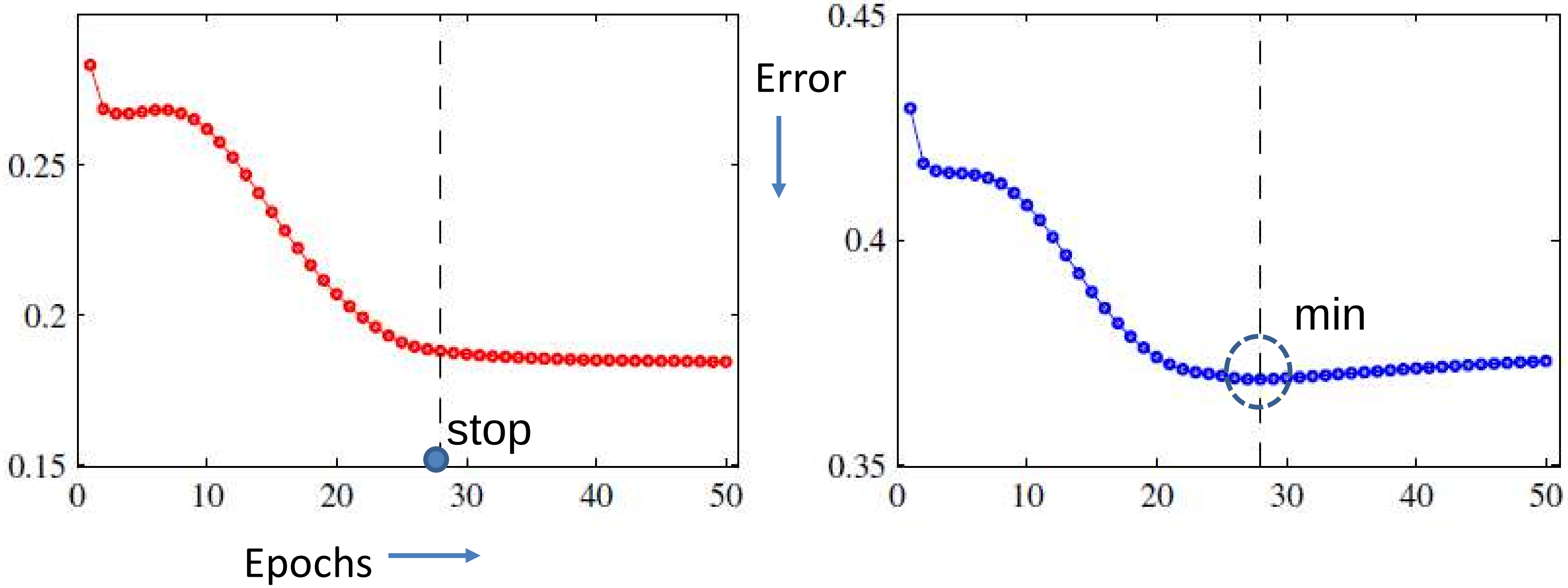


In L1 regularization (Lasso), the additional term added to the loss function is the sum of the absolute values of the weights. This encourages sparsity in the weights, effectively shrinking some of them to zero. $J_{L1}(\theta) = J(\theta) + \lambda\sum_{i=1}^{n}|\theta_i|$

Where, λ as the regularization parameter, ϑ as the vector of weights of the Network.

Regularization: Early Stopping

- Early stopping monitors the performance of the model on a validation set and stops training when the performance starts to degrade, thus preventing the model from overfitting to the training data.
- Example sinusoidal dataset



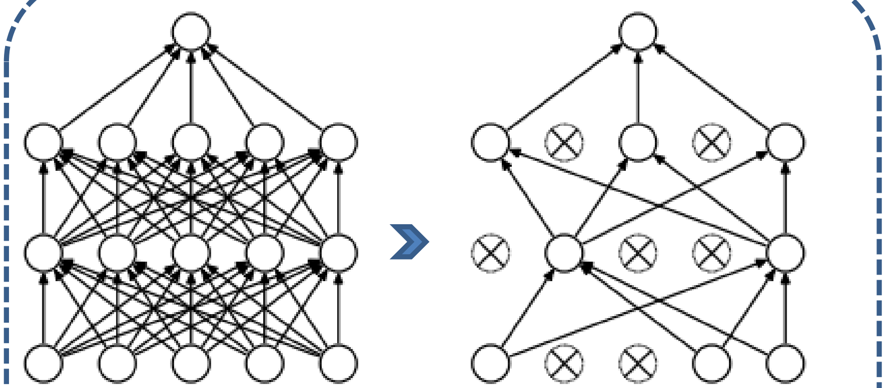
(Training set error)

(Validation set error)

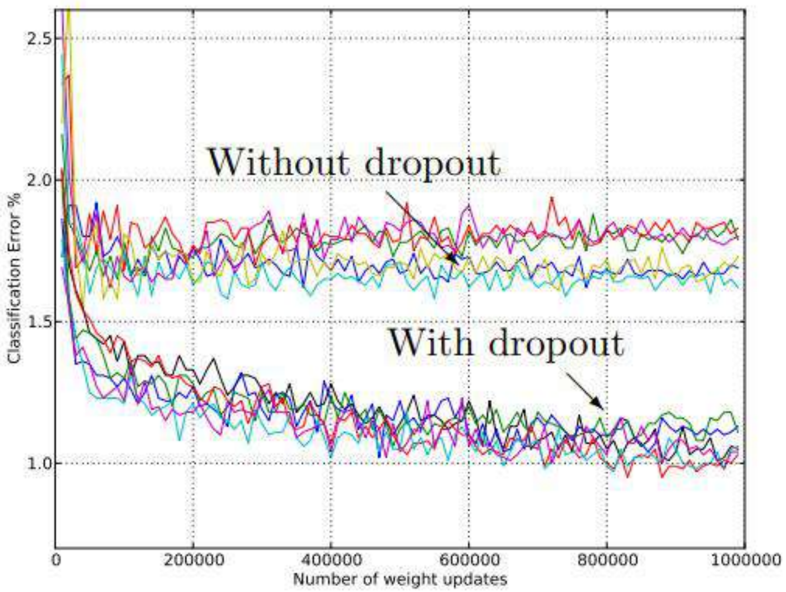
Img. Source: Bishop text

Stochastic Regularization: Dropout

- Drop out each individual unit with some probability ρ (usually ρ = 1/2) by setting its activation to ‘0’.
- The key idea behind dropout is to prevent overfitting by adding noise to the network during training.



(Standard Neural Network) (After applying dropout)



(Img. Source: Nitish Srivastava, et al., Journal of Machine Learning Research, 15, 2014)

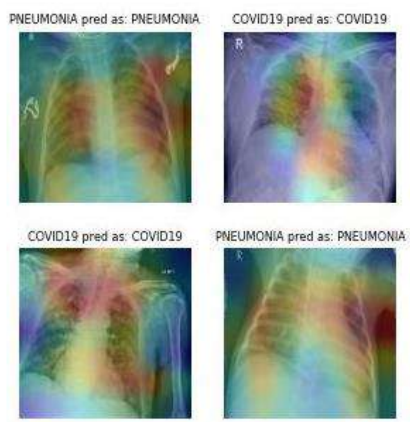
During inference (testing or prediction), dropout is typically turned off, and the full network is used. However, the weights are usually scaled by ‘1-ρ’ during inference to account for the fact that more units were active during training.

Regularization: Data Augmentation

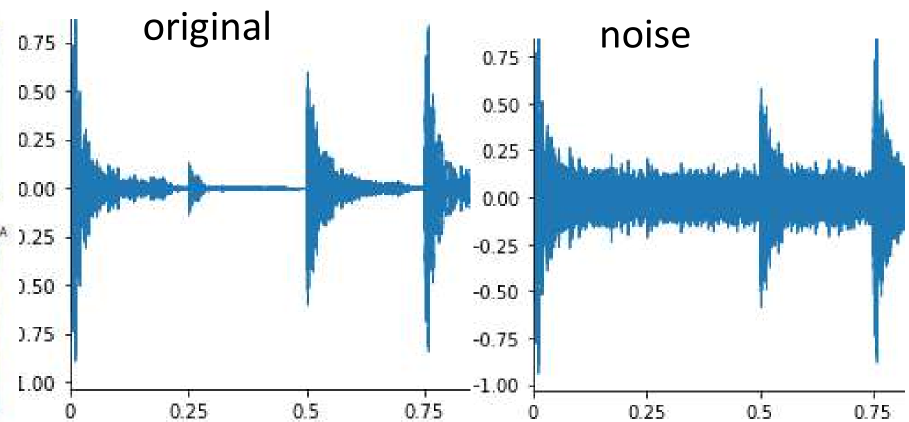
- Data augmentation acts as a form of regularization by introducing additional variations and diversity into the training dataset.
- A technique used to artificially increase the size of a training dataset by applying various transformations to the existing data samples.
- Random rotation, Random scaling, Random cropping, Horizontal or vertical flipping, Adding noise (e.g., Gaussian noise), Changing brightness, contrast, or saturation

Applications

(Self driving cars)



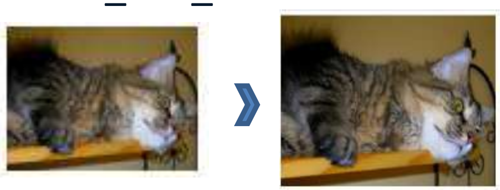
(Healthcare)



(Automatic Speech Recognition)

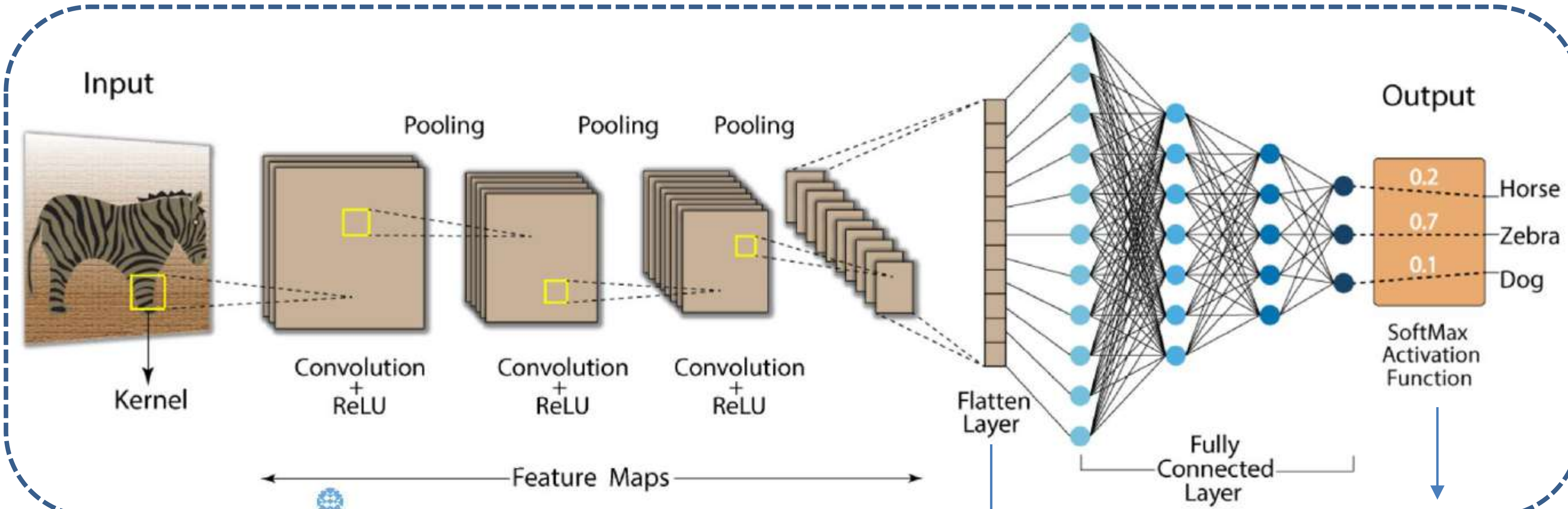
Img. Source: https://www.datacamp.com/

```
resize_and_rescale=keras.Sequential([ layers.Resizing(IMG_SIZE, IMG_SIZE), layers.Rescaling(1./255)])
```



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.

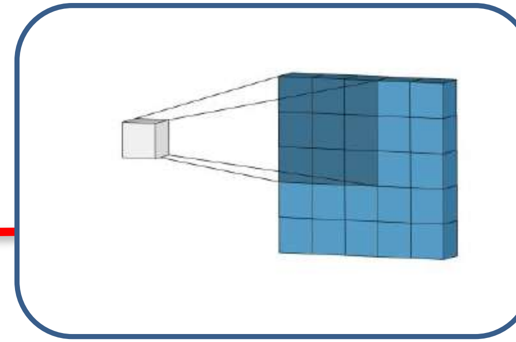


(Img. Source: https://developersbreach.com/)

Spatial structure to 1-D vector

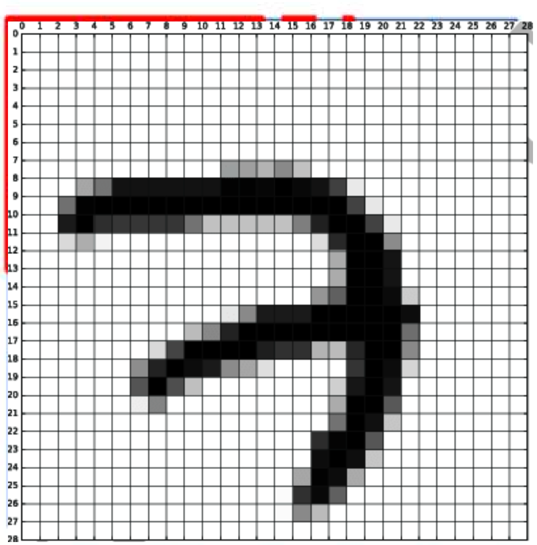
$$p_i(\boldsymbol{x}) = \frac{\exp(f_i(\boldsymbol{x}))}{\sum_{j=1}^{N} \exp(f_j(\boldsymbol{x}))}$$

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



Original image (6X6)

Convolution Operation

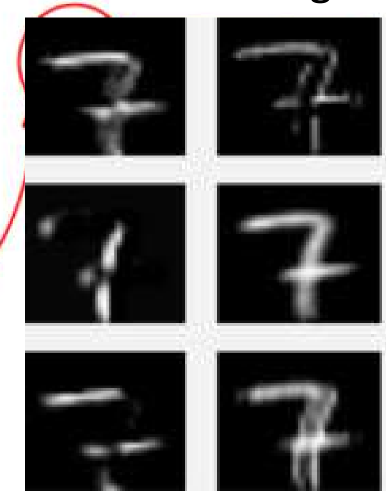
Horizontal Sobel Kernel

-1	0	1
-2	0	2
-1	0	1

Kernel (3X3)

Vertical Sobel Kernel

Corners or edges



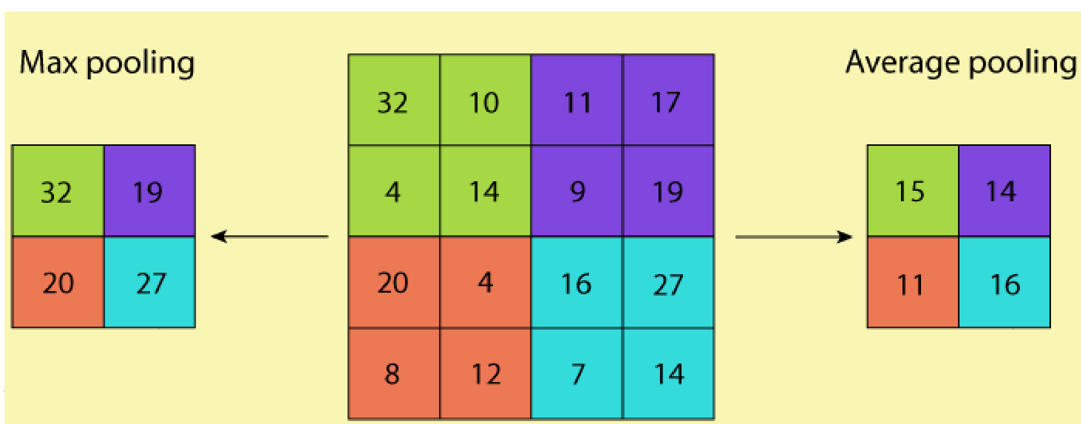
CNN: Pooling (or Sub-sampling) Layer

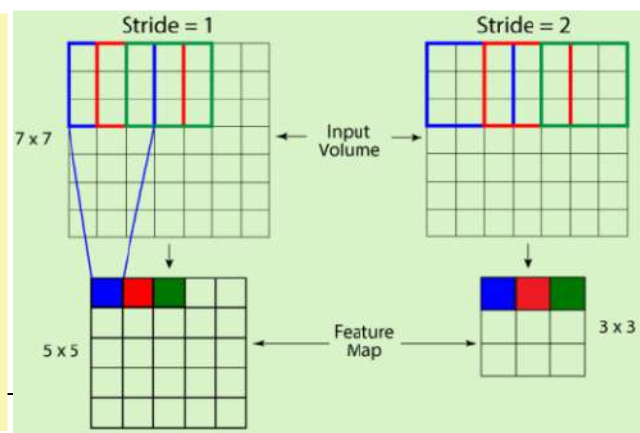
Spatial Invariance: Pooling layers aggregate information from local neighborhoods of the input feature map, which helps in creating spatial invariance.

→ 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.

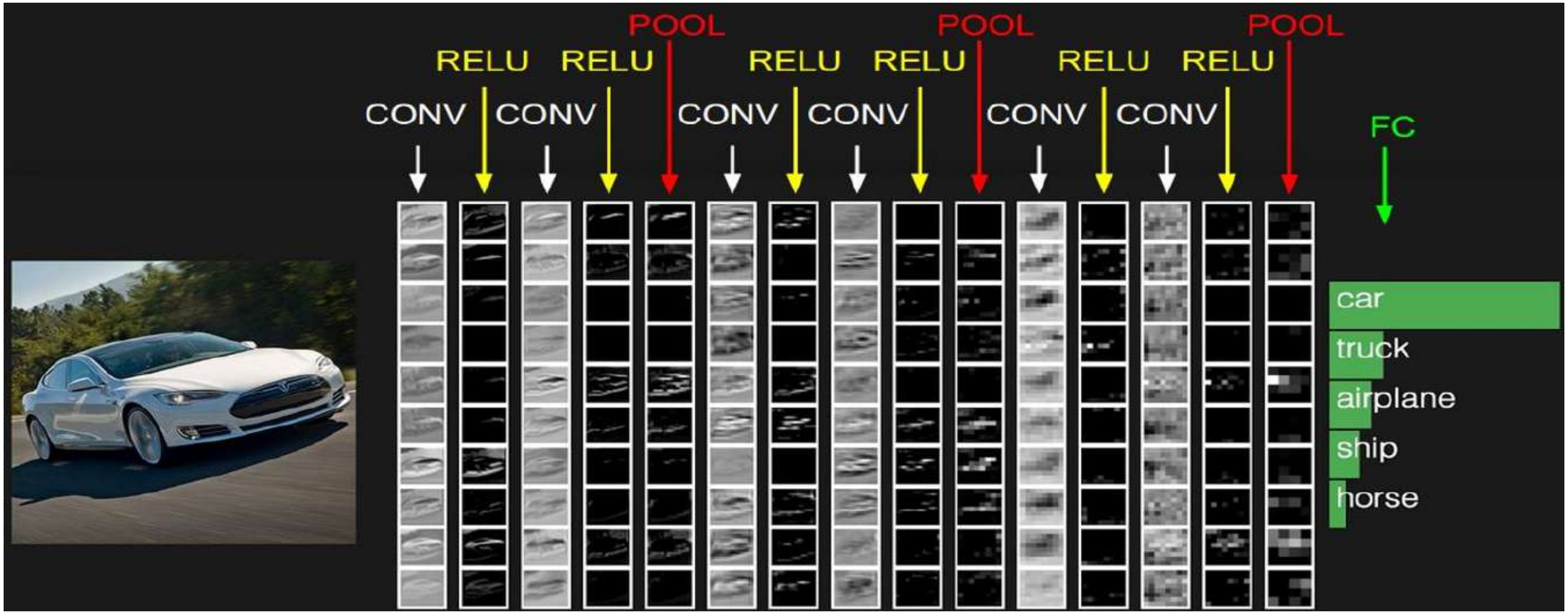
Local feature detection: Salient features within the local neighborhood is identified.





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

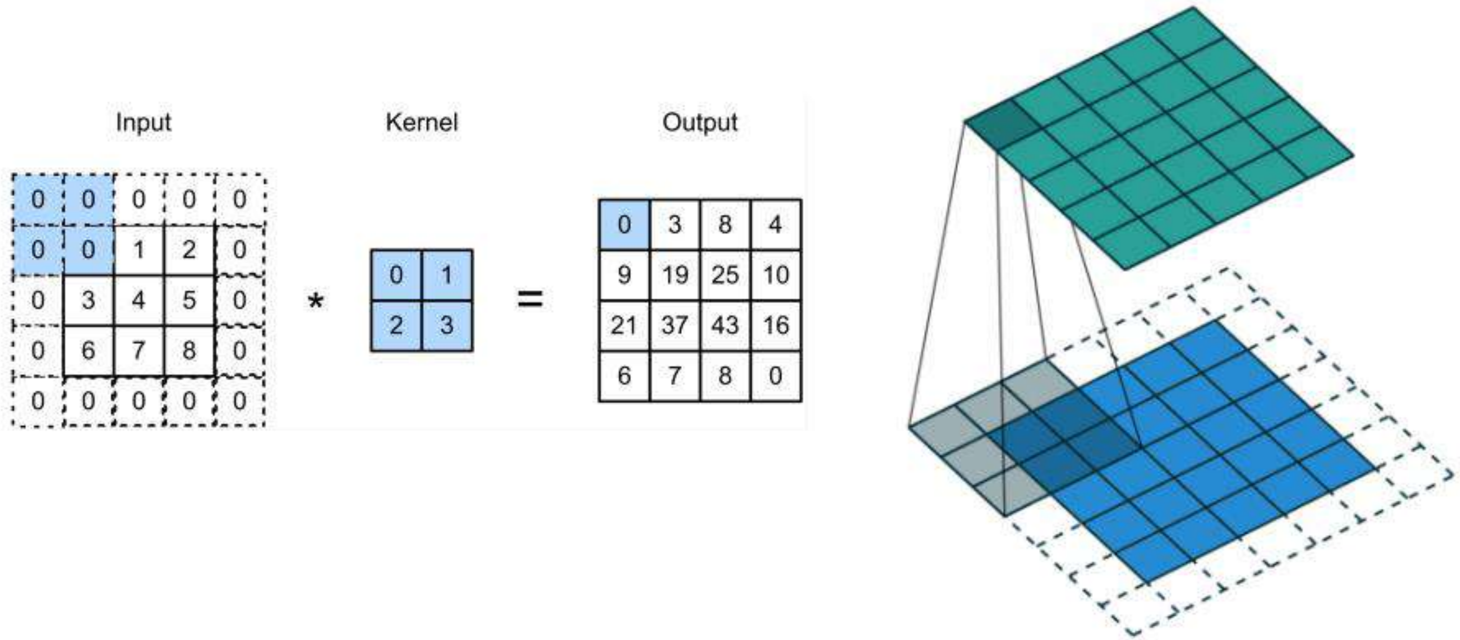
Classifying an Image: An example



Acknowledgement: Md Mahin's presentation.

CNN: Padding

- Padding in CNNs refers to the process of adding additional pixels around the input image before applying convolution operations.

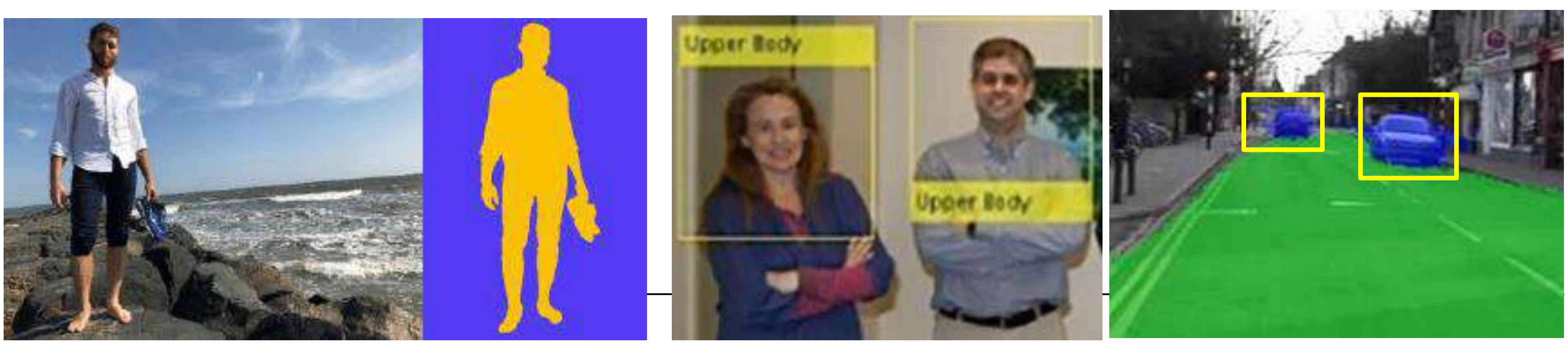


- It controls the spatial dimensions of feature maps and prevents information loss at the borders of the image.

Img. Source: Ilias Diakonikolas's class presentation, UW Madison

Other types of Convolutional Networks: FCN, SSMD

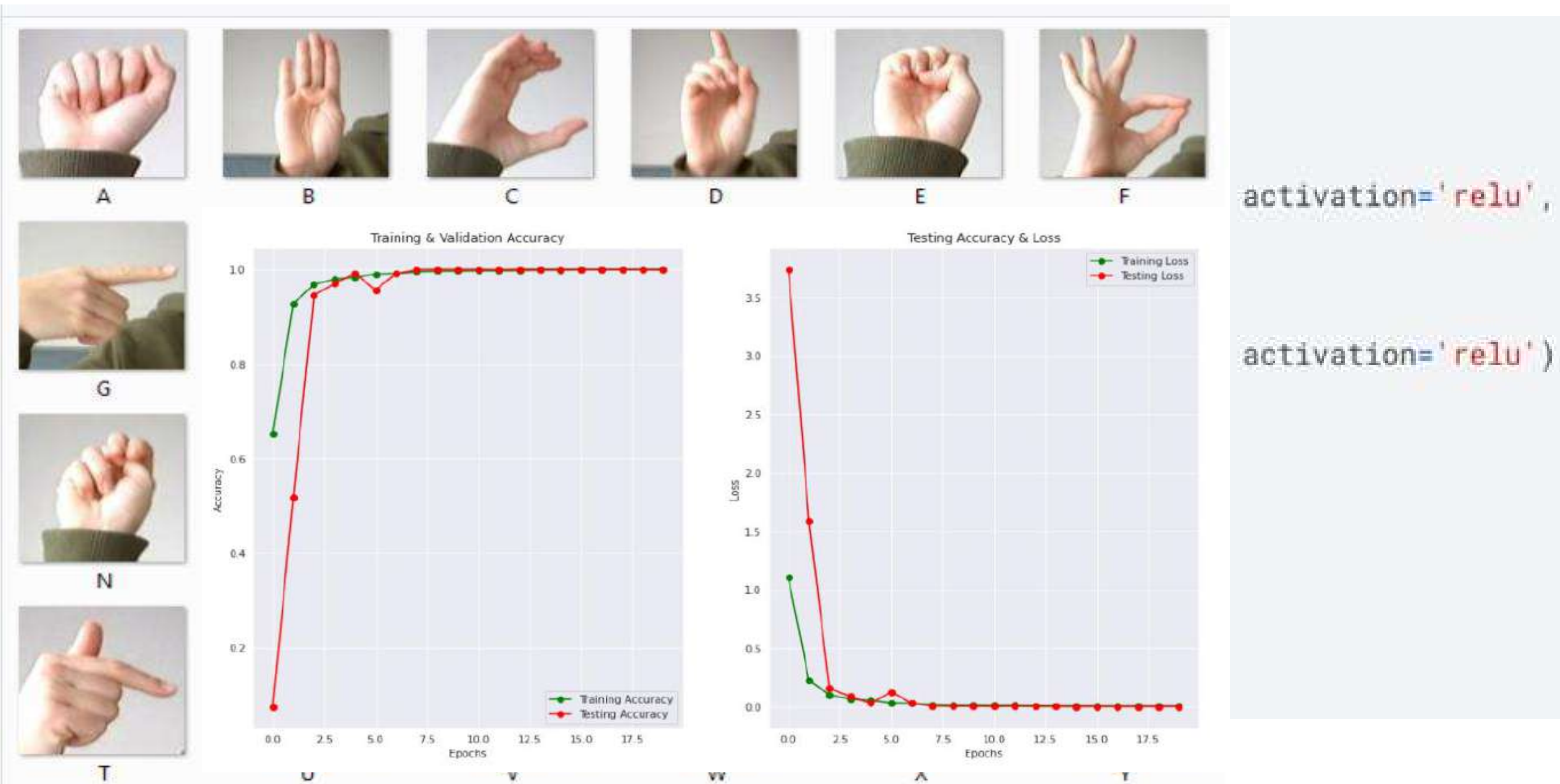
- Semantic segmentation, where the goal is to assign a class label to each pixel in the input image, the fully connected layer is not well-suited as they do NOT preserve spatial information.
- In Fully Convolutional Networks (FCNs), the final fully connected layer of the traditional CNN is replaced with **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

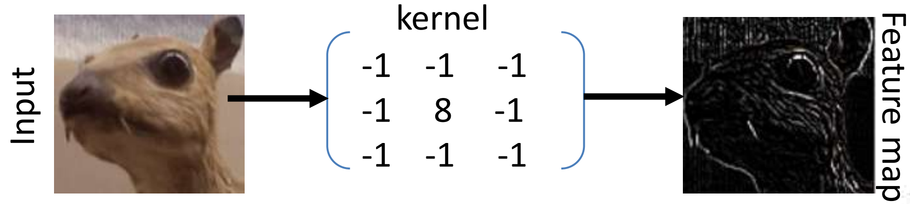


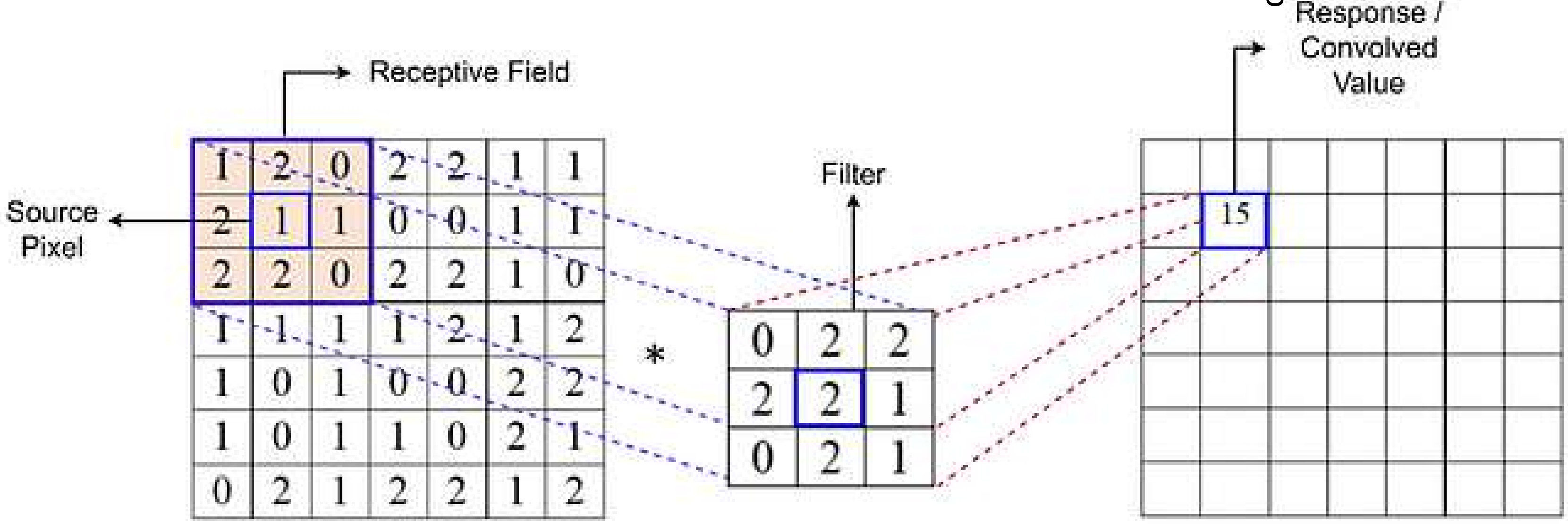
Submission deadline: 30.04.2024

Many others: ResNet, AlexNet, ImageNet

Recap: Convolution in CNNs

$$(x * f)[n] = \sum_{a=-M/2}^{M/2} x[n-a]\, f[a]$$





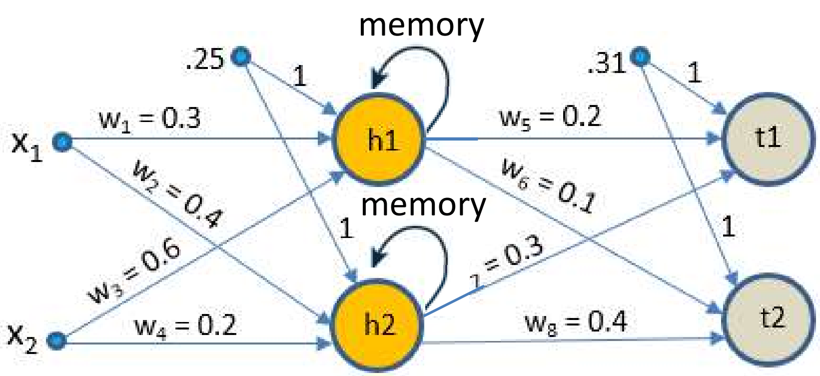
Convolution: A linear operation that computes the dot product of the local receptive field and the filter matrix.

It finds out the correction between the filter/ kernel with different parts of the image (when it slides over it), thereby learns important patterns or features.

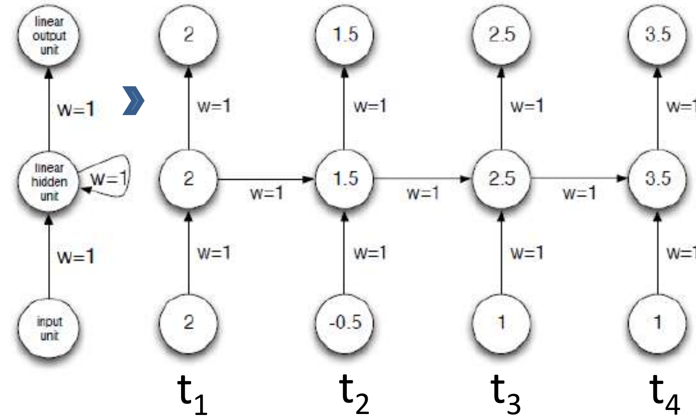
Recurrent Neural Networks (RNNs)

- 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.

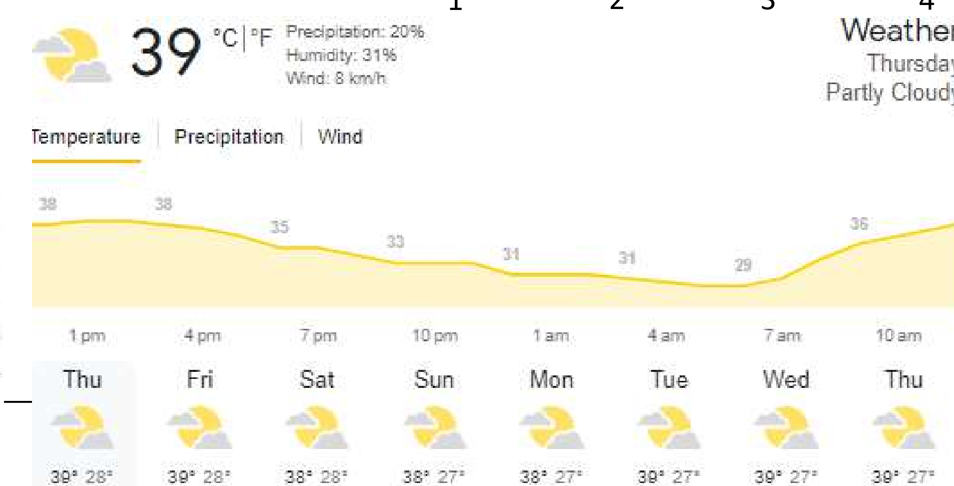
R
N
N



f
u
n
?

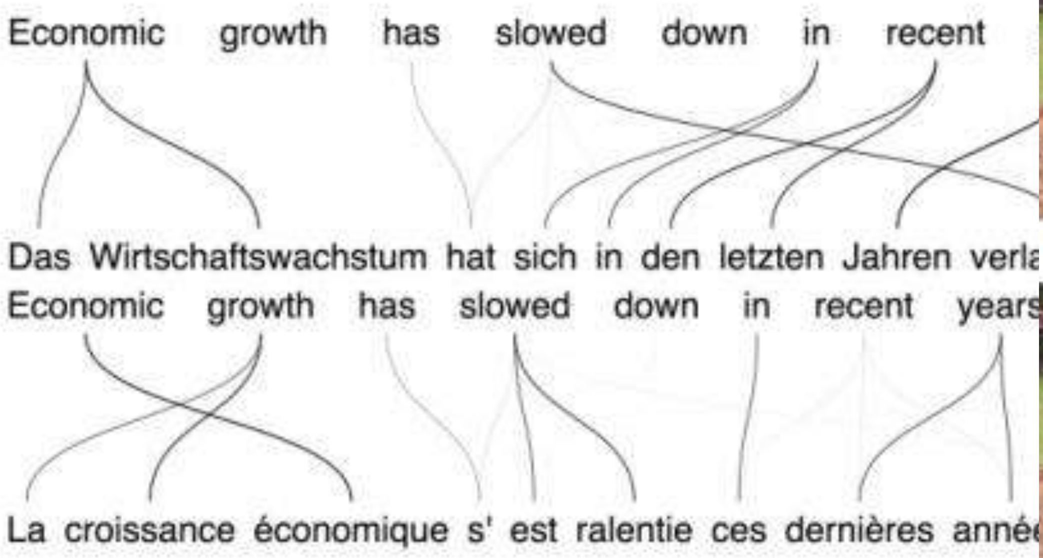


Applications



More Applications…

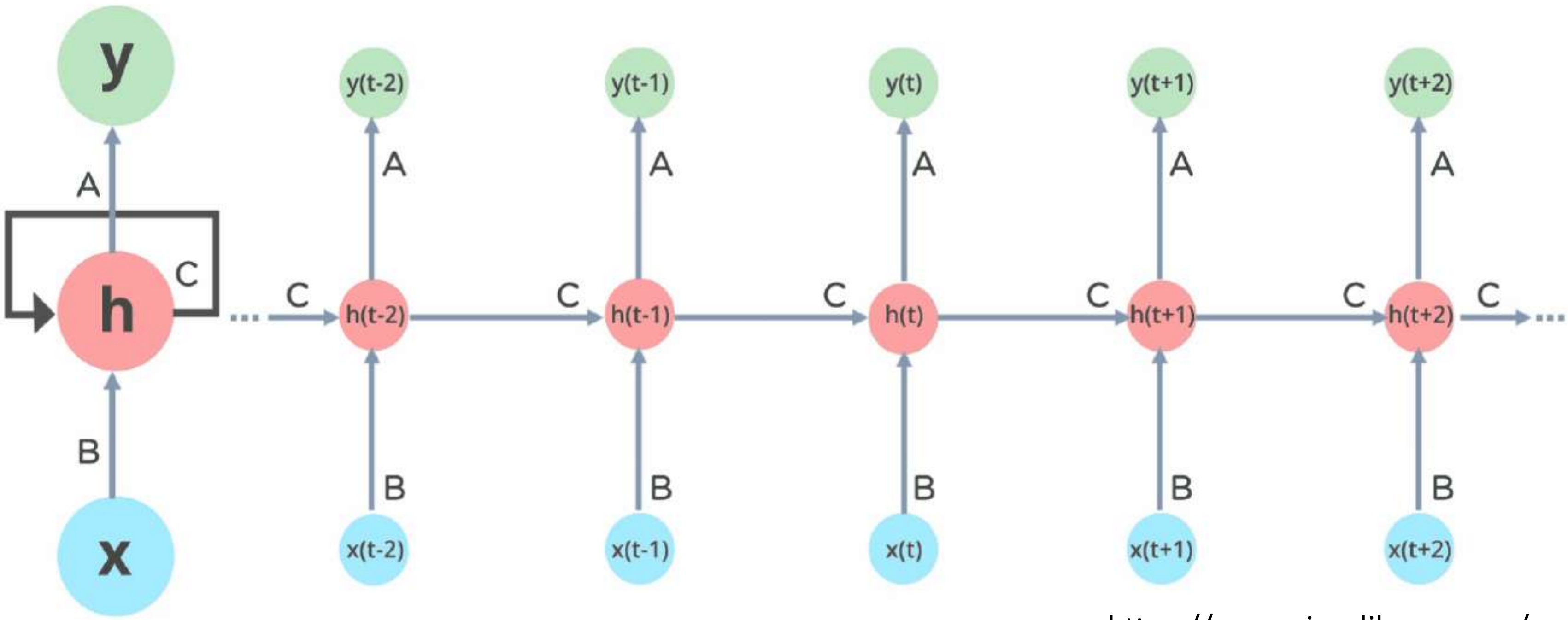
Machine Translation



Video Classification

Modelling sequential data: RNN

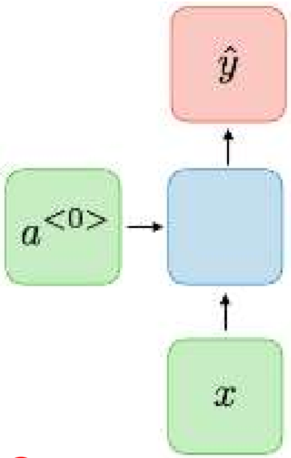
- A hidden state captures information about previous inputs. State is plugged back into itself.
- The hidden state is updated recurrently based on the current input and the previous hidden state.



https://www.simplilearn.com/

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

RNN Architectures: One-to-One

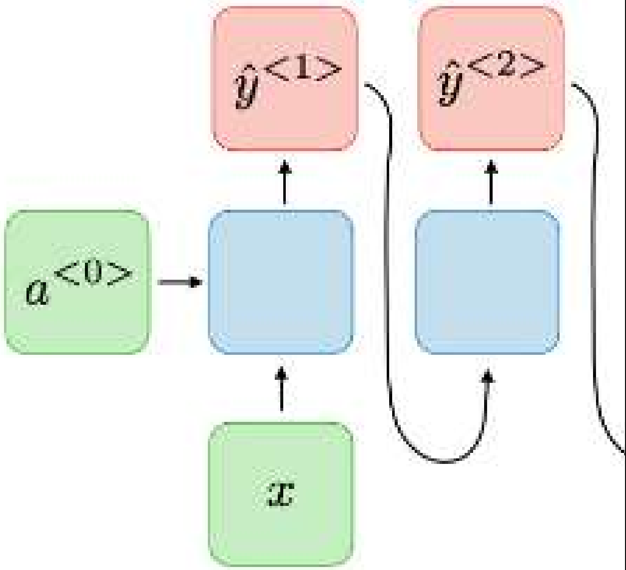


One-to-one: Traditional NN

without considering temporal dependencies between reviews.

```
# Define the RNN model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=16, i
    tf.keras.layers.LSTM(32),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

# Train the model
model.fit(padded_sequences, labels, epochs=10, verbose=1)

# Inference
test_review = "This film was excellent!"
test_sequence = tokenizer.texts_to_sequences([test_review])
padded_test_sequence = pad_sequences(test_sequence, maxlen=max_length, padding='post'
prediction = model.predict(padded_test_sequence)

# Output the predicted sentiment
if prediction > 0.5:
    print("Positive sentiment")
else:
    print("Negative sentiment")
```

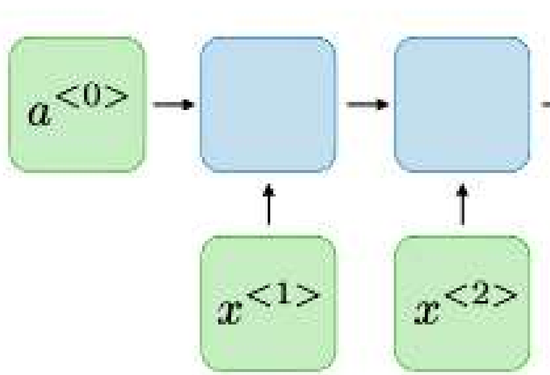
RNN Architectures: One-to-Many



One-to-Many: Music generation

```
# 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)
```

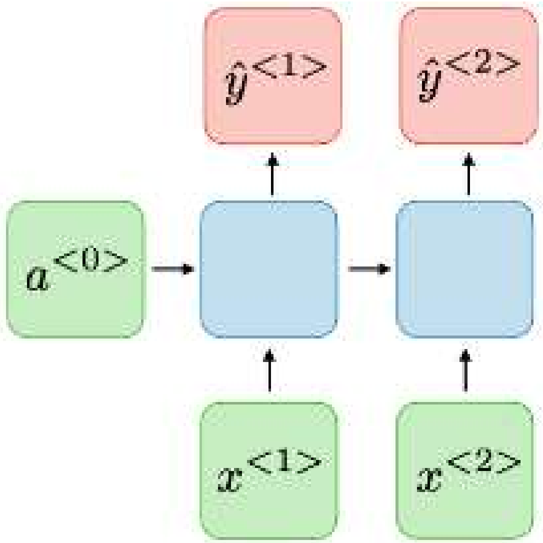
RNN Architectures: Many-to-one

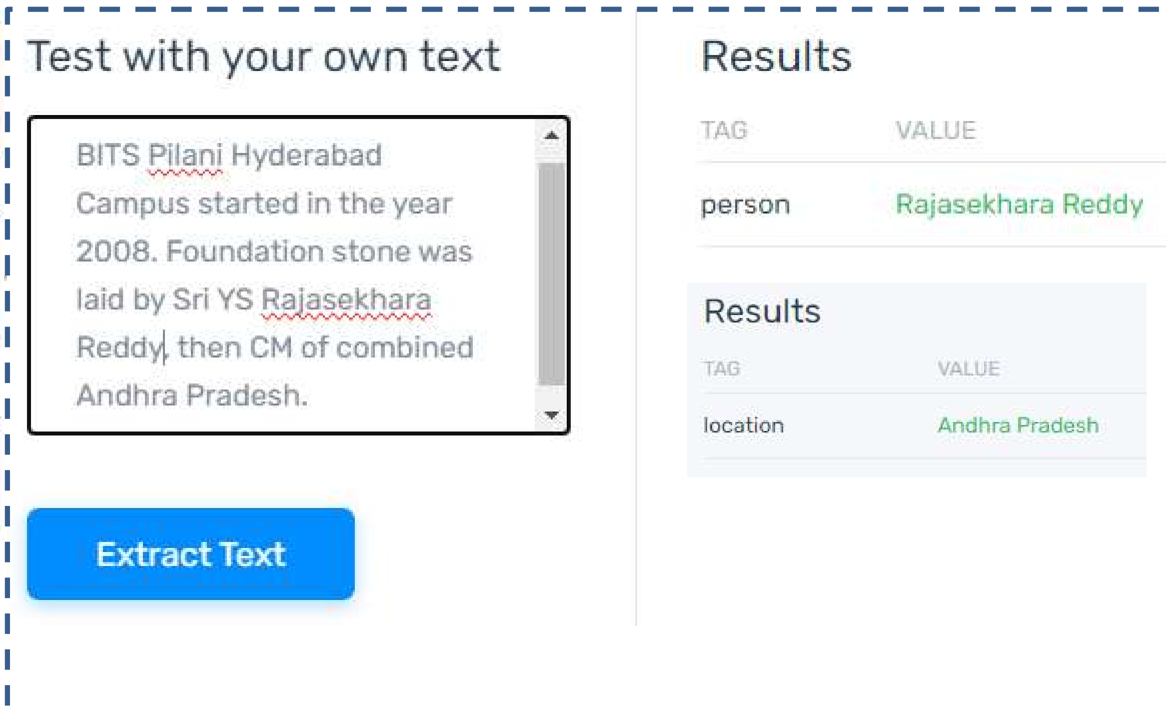



```
# Build the RNN model
model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(max_words, 16, input_length=max_sequence_length),
    tf.keras.layers.LSTM(32),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

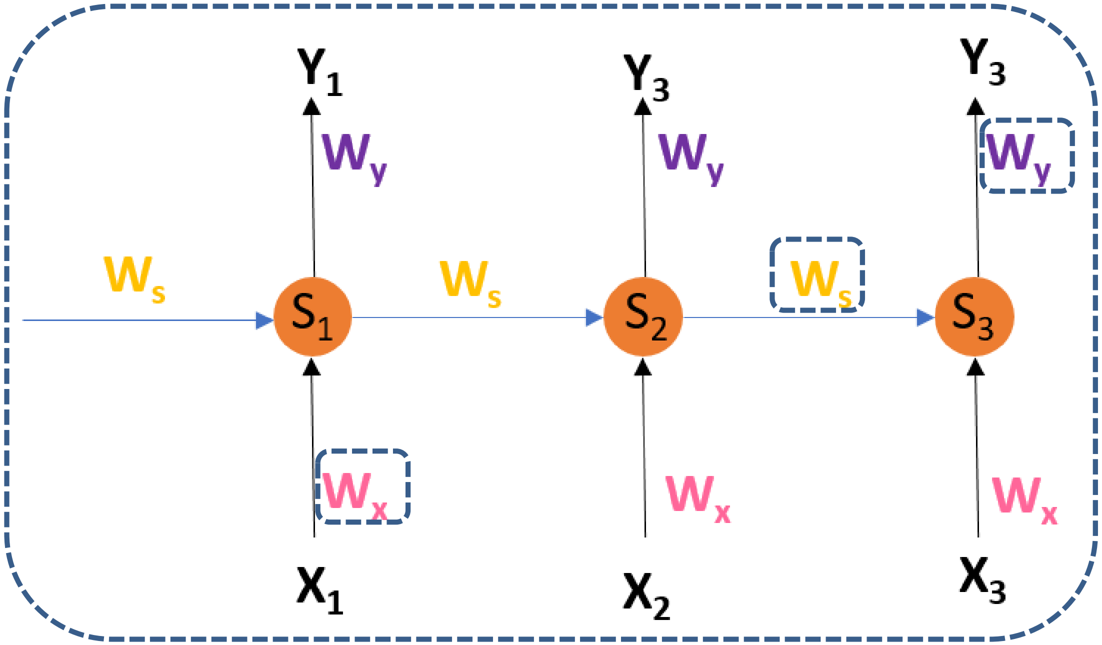
RNN Architectures: Many-to-Many





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:

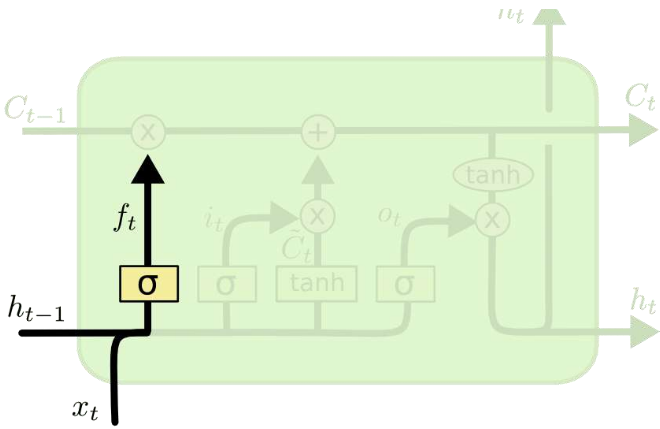
$$\frac{\partial E_3}{\partial W_s} = \underbrace{\left\{\frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial Y_3}{\partial S_3} \cdot \frac{\partial S_3}{\partial W_s}\right\}}_{E_3 \text{ --- } Y_3 \text{ --- } S_3 \text{ --- } W_s} + \underbrace{\left\{\frac{\partial E_3}{\partial Y_3} \cdot \frac{\partial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

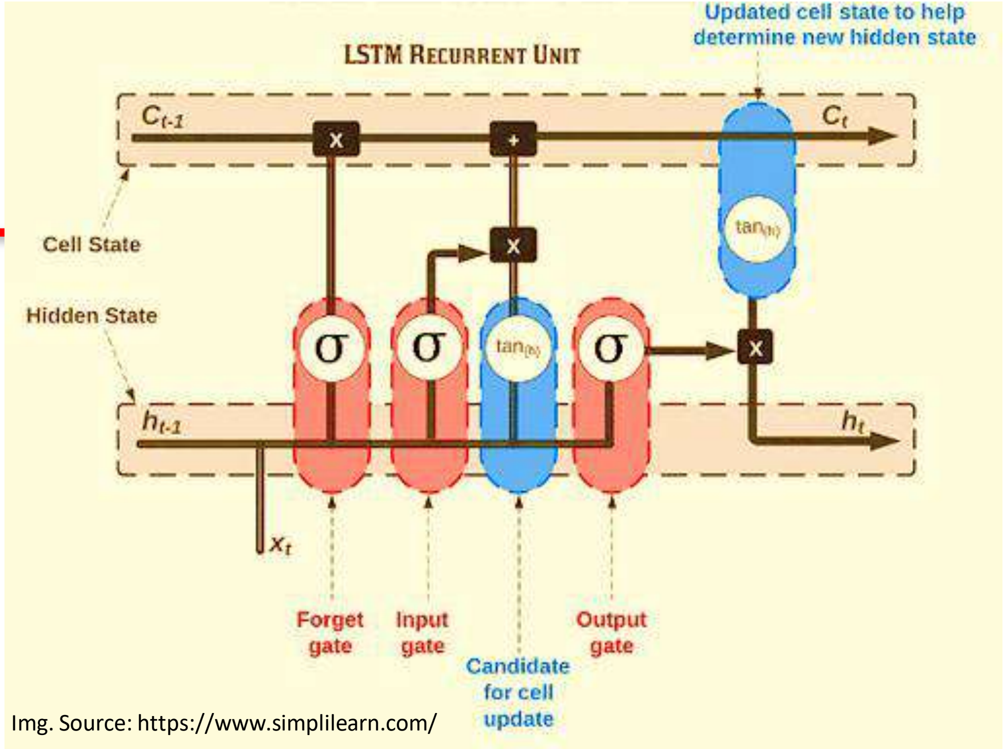
For W_x:

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

LSTM: RNN

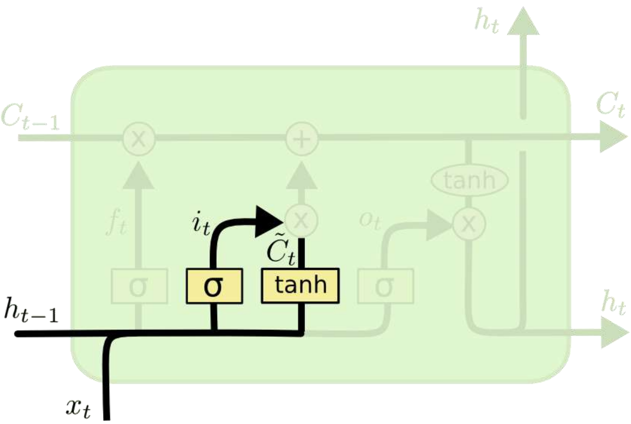
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$





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

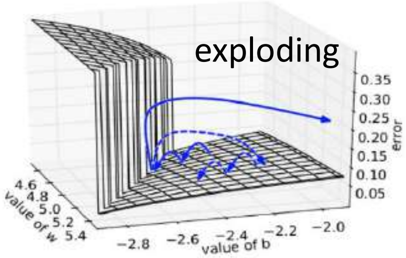
What info. is to be thrown away? → Forget gate (ex: gender of old subject)



What new info. is to be stored?

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

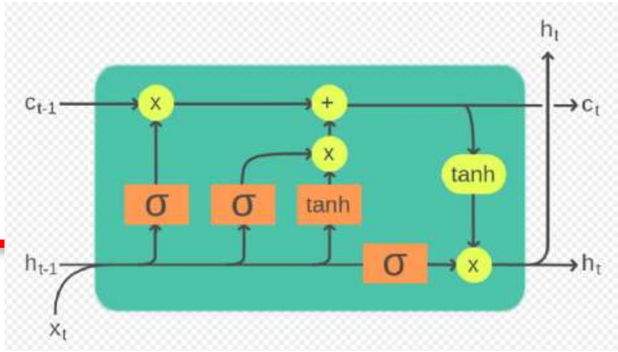
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

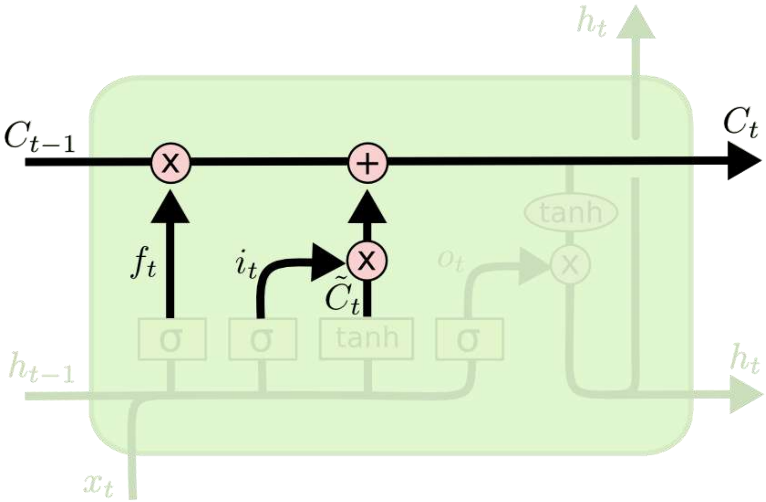


May be gender of the new subject.

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

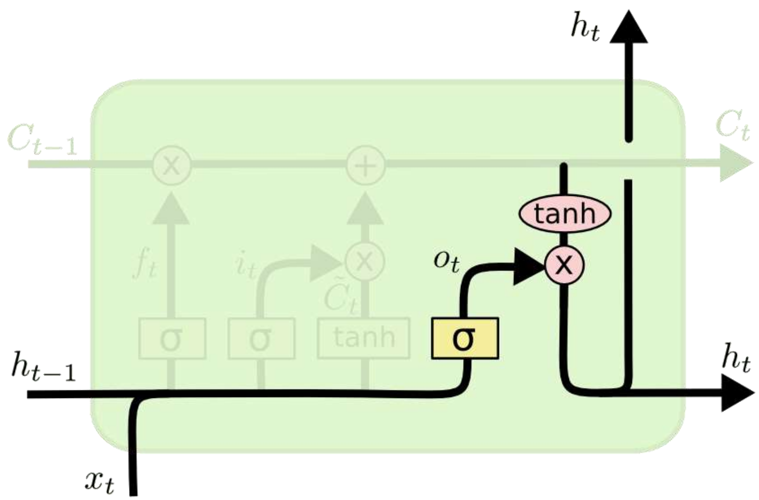
Continued...





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

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

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

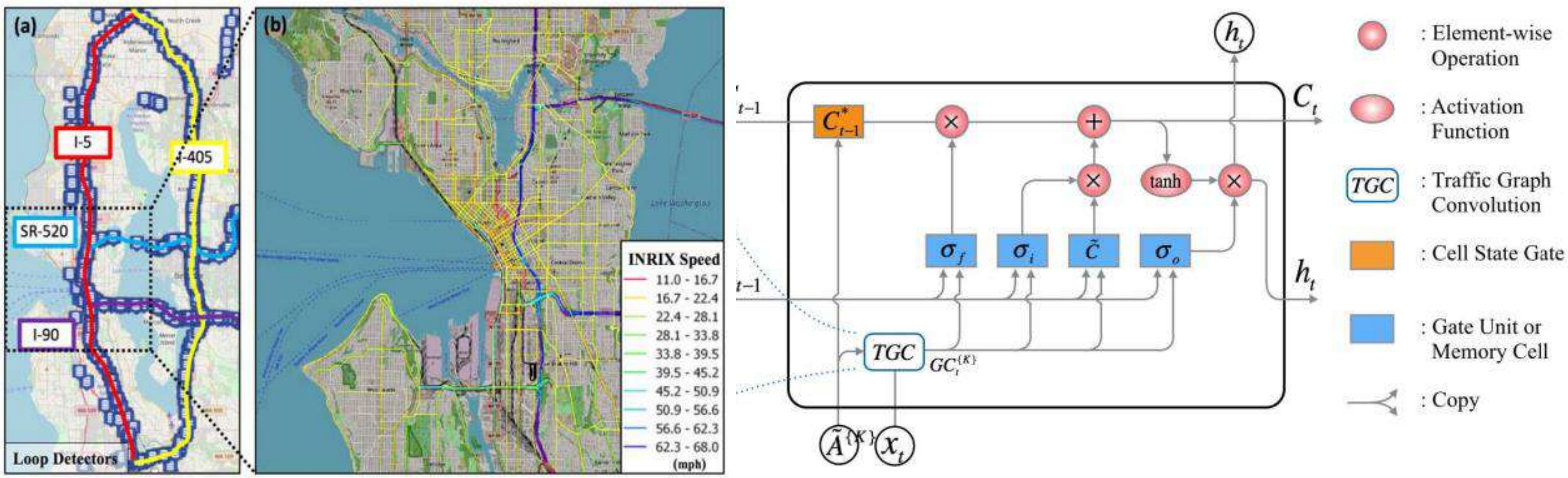
Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 21, NO. 11, NOVEMBER 2020

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.



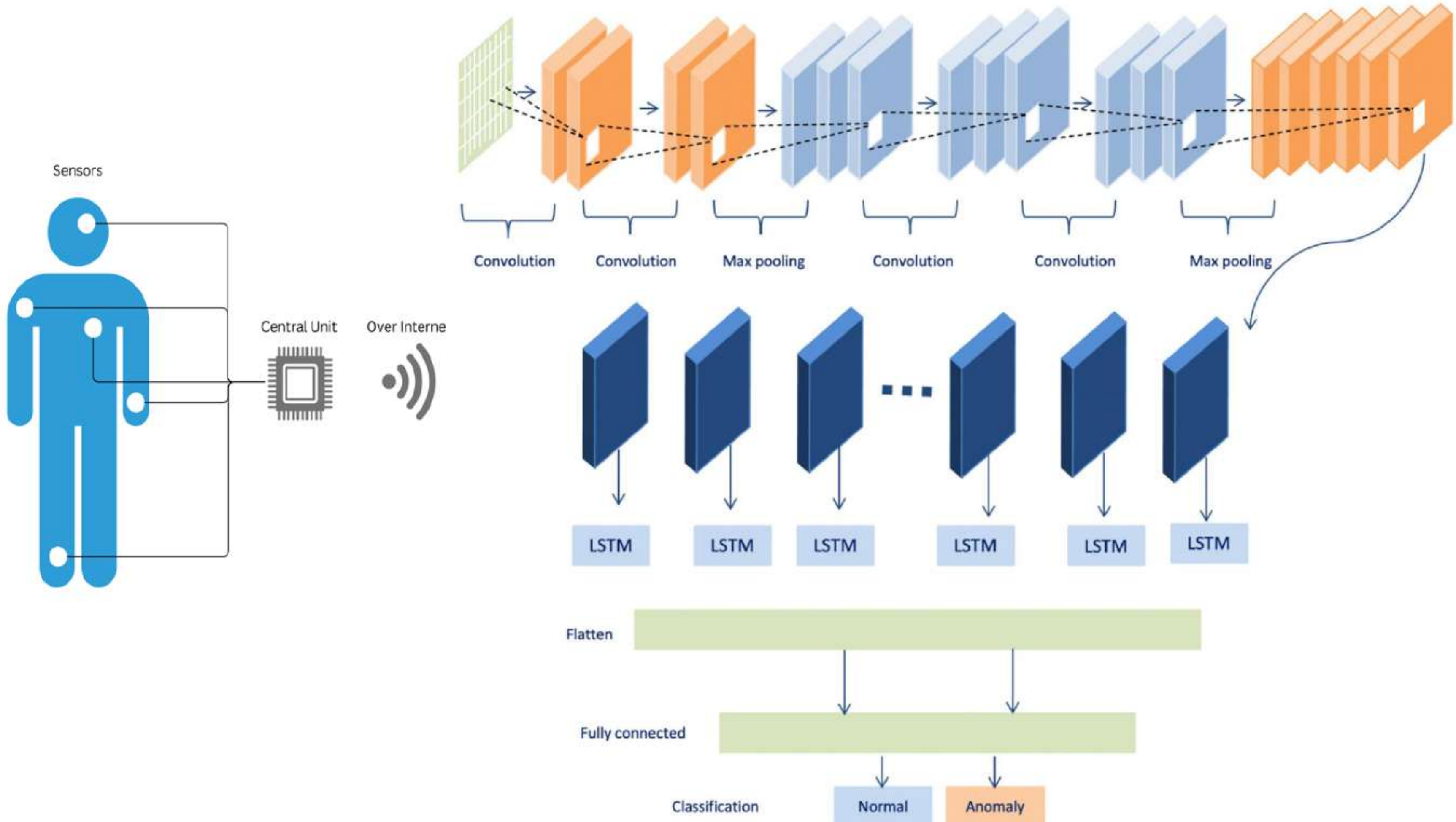
FFR: Free Flow Reachability (i.e vehicle speed) info, Ã: neighbourhood information, W: weights. Convolution layer is within the cell. Spatiotemporal dataset.

sensors 2022 MDPI

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,*]



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

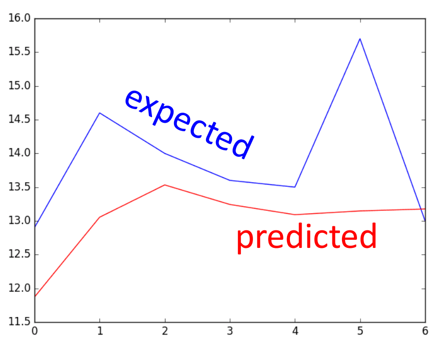
Current value

Previous value

Where, Φ_0, Φ_1 are coefficients to be estimated. ξ_t is error term at time 't'.

AR(1): Order 1

What would be AR(2), AR(3),…



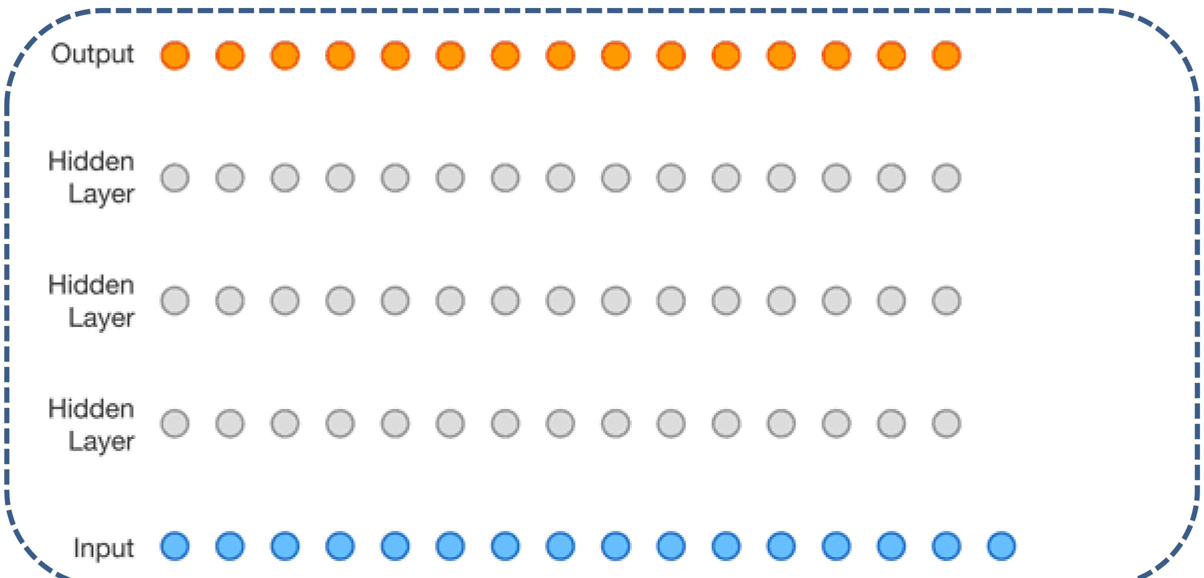


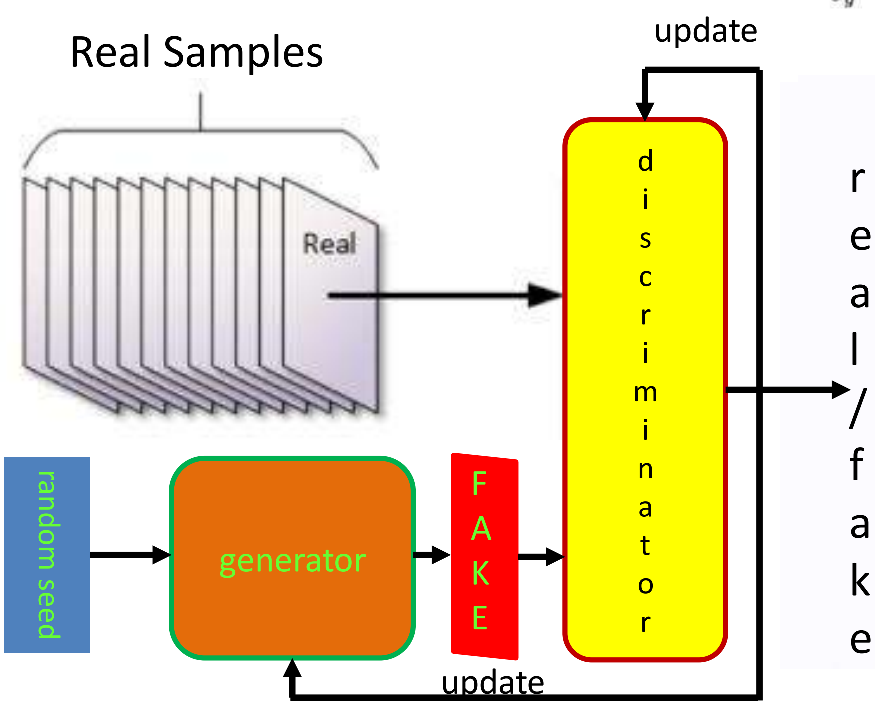
Image source: WaveNet (Google DeepMind)

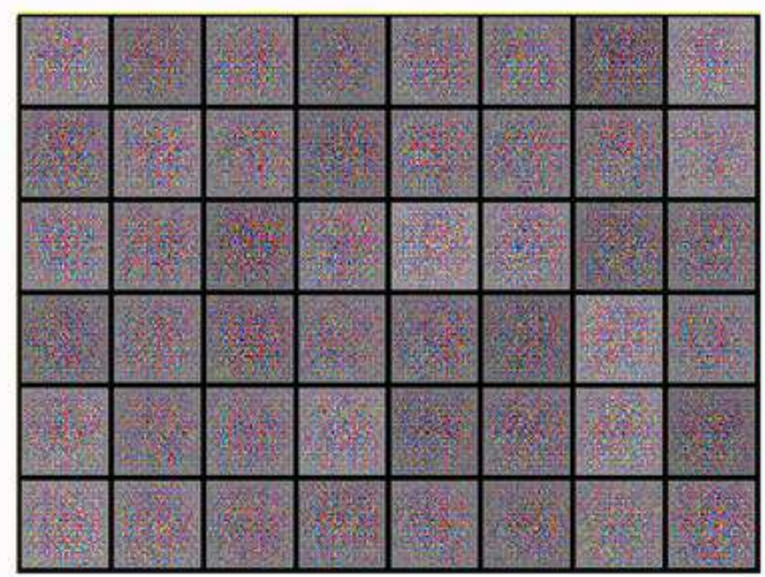
Probabilistically, Autoregression can be expressed as: $p(x) = \prod_{i=1}^{n} p(x_i | x_1, x_2, \ldots x_{i-1}) = \prod_{i=1}^{n} p(x_i | x_{<i})$

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)…

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$





Source: https://theaisummer.com

(Deep Convolutional GAN: Better Quality)

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

Thank You!