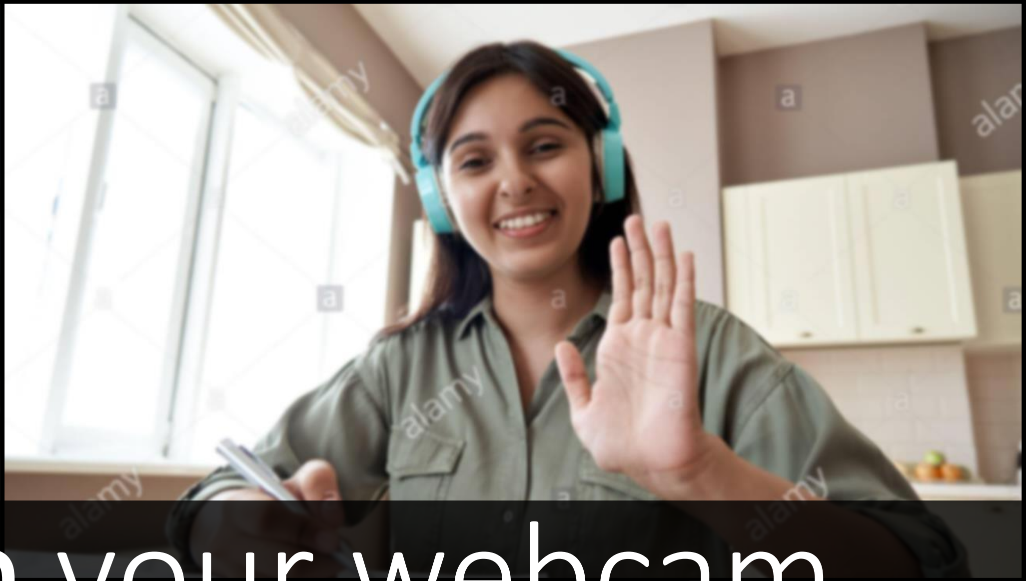
Deep Learning

10B

CS 3244
Machine Learning

NUS | Computing
National University of Singapore

Please **turn on** your webcam



Mystery Student

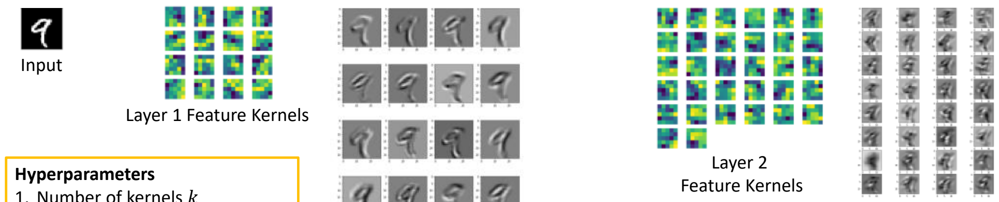
Recap from W10

Convolutional Layer: Feature Kernels & Feature Maps

$k^{[l-1]} = c^{[l]}$
filters from previous layer $k^{[l-1]}$ is equal to #channels into current layer $c^{[l]}$

$$\boldsymbol{X}^{[0]} \rightarrow g^{[1]}\left(\boldsymbol{W}^{[1]} * \boldsymbol{X}^{[0]}\right) = \boldsymbol{A}^{[1]} \xrightarrow{\text{Pooling}} g^{[2]}\left(\boldsymbol{W}^{[2]} * \boldsymbol{X}^{[1]}\right) = \boldsymbol{A}^{[2]}$$

1 channel | 16 filters | 16 channels | 32 filters | 32 channels



Hyperparameters
1. Number of kernels k
2. Kernel size $\boldsymbol{\kappa}$
3. Padding $\boldsymbol{p}$
4. Stride $\boldsymbol{s}$

Chosen manually, or automatically with hyperparameter tuning

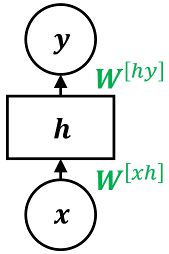
Kernels are learned *automatically* through **weight updates**.

Interpretability: do you know what these kernel mean?

Image credit: https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-networks-e1462340518e

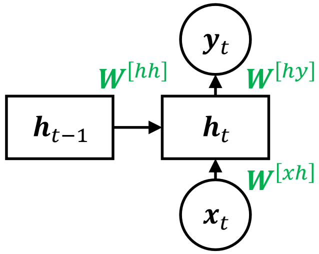
RNN Weights

Question: Do these weights change for different time t?



Feedforward Neural Network

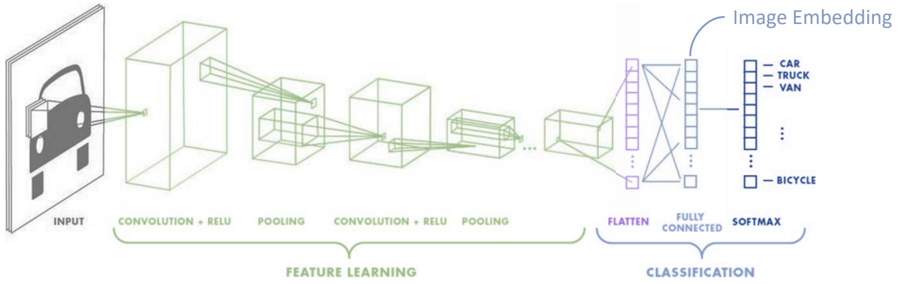
$$\boldsymbol{y} = g^{[y]}\left(\left(\boldsymbol{W}^{[hy]}\right)^\top \boldsymbol{h}\right)$$
$$\boldsymbol{h} = g^{[h]}\left(\left(\boldsymbol{W}^{[xh]}\right)^\top \boldsymbol{x}\right)$$



Recurrent Neural Network

$$\boldsymbol{y}_t = g^{[y]}\left(\left(\boldsymbol{W}^{[hy]}\right)^\top \boldsymbol{h}_t\right)$$
$$\boldsymbol{h}_t = g^{[h]}\left(\left(\boldsymbol{W}^{[xh]}\right)^\top \boldsymbol{x}_t + \left(\boldsymbol{W}^{[hh]}\right)^\top \boldsymbol{h}_{t-1}\right)$$
$$\boldsymbol{h}_t = g^{[h]}\left(\left(\boldsymbol{W}^{[xh]} \oplus \boldsymbol{W}^{[hh]}\right)^\top (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

Convolutional Neural Network



Key concepts

❶ Learn Spatial Feature
- Series of multiple convolution + pooling layers
- Progressively learn more diverse and higher-level features

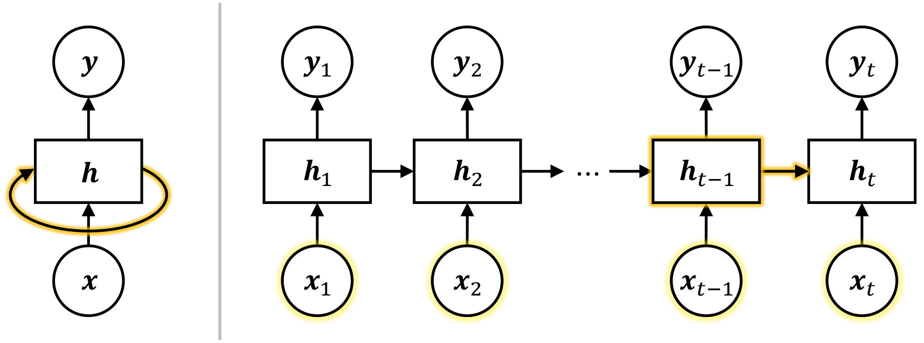
❷ Flattening
- Convert to fixed-length 1D vector

❸ Learn Nonlinear Features
- With fully connected layers (regular neurons)
- Learns nonlinear relations with multiple layers

❹ Classification
- Softmax ≔ Multiclass Logistic Regression
- Feature input = image embedding vector (typically large vector)

Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Neurons with Recurrence



$$\hat{\boldsymbol{y}} = g^{[y]}\left(g^{[h]}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})\right)$$

Recurrent Neural Network (RNN)

$$\hat{\boldsymbol{y}} = g^{[y]}(\boldsymbol{h}_t)$$
$$\boldsymbol{h}_t = g^{[h]}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

What are the Kernel Size, Stride, Padding?

$\{height \times width\}$ $\dim \boldsymbol{x} = \{2 \times 6\}$ $\dim \boldsymbol{y} = \{4 \times 3\}$

$$\boldsymbol{W} = \begin{pmatrix} -1 & 0 \\ -1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$\boldsymbol{x} = \begin{array}{l} h_p/2 \{ \\ \\ \begin{pmatrix} 9 & 9 & 9 & 3 & 3 & 4 \\ 9 & 9 & 3 & 5 & 5 & 8 \end{pmatrix} \\ \\ h_p/2 \{ \end{array} \quad w_s$$

$$\boldsymbol{y} = \boldsymbol{W} * \boldsymbol{x} = \begin{pmatrix} 0+0+9 & 0+0+3 & 0+0+4 \\ 0+0+9 & 0-6+5 & 0+1+8 \\ -9+0+0 & -9+2+0 & -3+3+0 \\ -9+0+0 & -3+0+0 & -5+0+0 \end{pmatrix}$$

Hyperparameters
- Kernel size $\boldsymbol{\kappa} = \{3 \times 2\}$
- Padding $\boldsymbol{p} = \{(2+2) \times 0\}$
- Stride $\boldsymbol{s} = \{1 \times 2\}$

Chosen manually, or automatically with hyperparameter tuning

$$\dim \boldsymbol{y} = \left\{\left(\frac{h_x + h_p - h_\kappa}{h_s} + 1\right) \times \left(\frac{w_x + w_p - w_\kappa}{w_s} + 1\right)\right\}$$

$$= \left\{\left(\frac{2+4-3}{1} + 1\right) \times \left(\frac{6+0-2}{2} + 1\right)\right\}$$

Lecture Schedule update

Week	Lecture (Mon)	Lecture (Thu)	Tutorial
11	Explainable AI	AMA	Deep Learning
12	Unsupervised Learning	Deepavali (Holiday)	Explainable AI + Deep Learning
13	AI Ethics	AMA	Unsupervised Learning
Exam	Wed 24 Nov @ 5-7pm		

Week 10C: Learning Outcomes

1. Understand how deep learning enables better model performance than shallow machine learning
2. Explain how CNNs and RNNs are different from feedforward neural networks
3. Appropriately choose and justify when to use each architecture
4. Explain how to mitigate training issues in deep learning

Week 10C: Lecture Outline

1. Deep learning motivation
2. Popular Architectures
    1. Convolutional Neural Networks
    2. Recurrent Neural Networks
3. Deep learning training issues

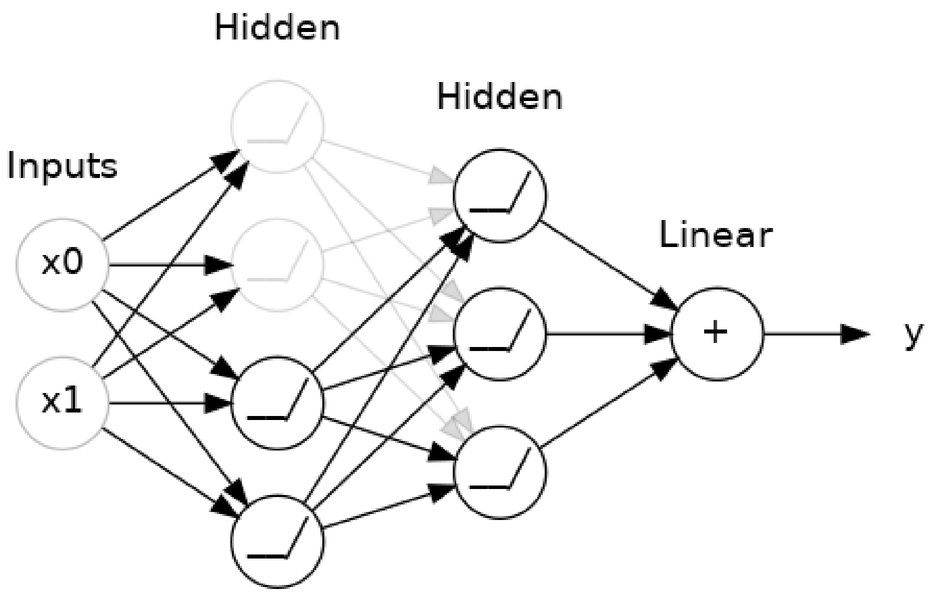
Deep Learning Training Issues

Deep Learning Training Issues

- Overfitting
- Saturating Gradient Problem
- Vanishing Gradient Problem

Overfitting in deep neural networks

- Recall: what is overfitting?
- Why can deep learning overfit?
  - Too many parameters!
- Mitigation?
  - Dropout
    - **Randomly "drop out"** some neurons during batch **training**
    - **Cannot propagate** through those neurons during training
    - Note: **all nodes** are still used for prediction

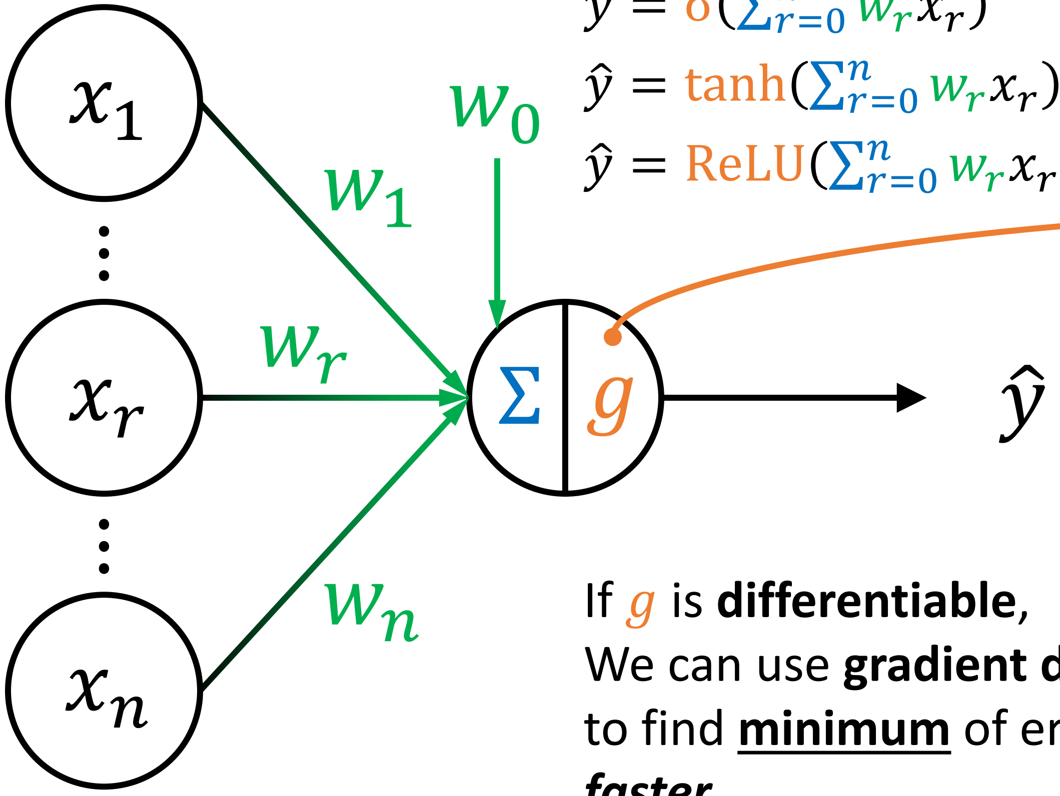


Further reading: https://towardsdatascience.com/12-main-dropout-methods-mathematical-and-visual-explanation-58cdc2112293

Deep Learning Training Issues

- Overfitting
- Saturating Gradient Problem
- Vanishing Gradient Problem

Differentiable Activation Functions



$\hat{y} = \mathrm{sgn}(\sum_{r=0}^{n} w_r x_r)$

$\hat{y} = \sigma(\sum_{r=0}^{n} w_r x_r)$

$\hat{y} = \tanh(\sum_{r=0}^{n} w_r x_r)$

$\hat{y} = \mathrm{ReLU}(\sum_{r=0}^{n} w_r x_r)$

If g is **differentiable**,
We can use **gradient descent**
to find **<u>minimum</u>** of error $\varepsilon = y - \hat{y}$
faster

Step

$$\mathrm{sgn}(x) = \begin{cases} +1 & z > 0 \\ -1 & z \leq 0 \end{cases}$$

Sigmoid

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

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

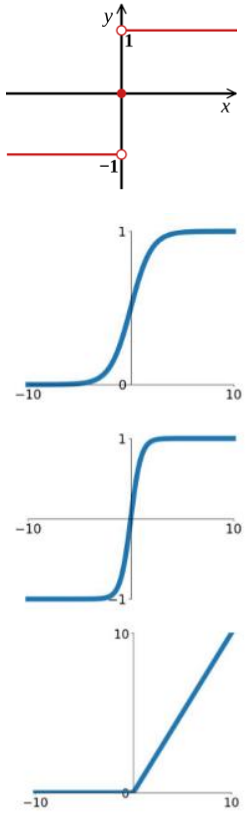


Image Credit:
https://miro.medium.com/max/1400/0*sIJ-gbjlz0zrz8lb.png

Gradient Descent Weight Update

$$\underbrace{\boldsymbol{W}}_{\text{New weight}} \leftarrow \overbrace{\boldsymbol{W}}^{\text{Old weight}} - \underbrace{\eta}_{\text{Learning Rate}} \overbrace{\nabla\varepsilon}^{\text{Direction of fastest error increase}}$$

MSE error

$$\varepsilon = \frac{1}{2}(\hat{y} - y)^2$$

Gradient of error

$$\nabla\varepsilon = \frac{\partial\varepsilon}{\partial\boldsymbol{W}} = \frac{\partial\varepsilon}{\partial\hat{y}}\underbrace{\frac{\partial\hat{y}}{\partial\boldsymbol{W}}}_{\frac{\partial f}{\partial\boldsymbol{W}}\frac{\partial g}{\partial f}}$$

Reference

$$\boldsymbol{a}^{[l]} = g^{[l]}(f^{[l]})$$

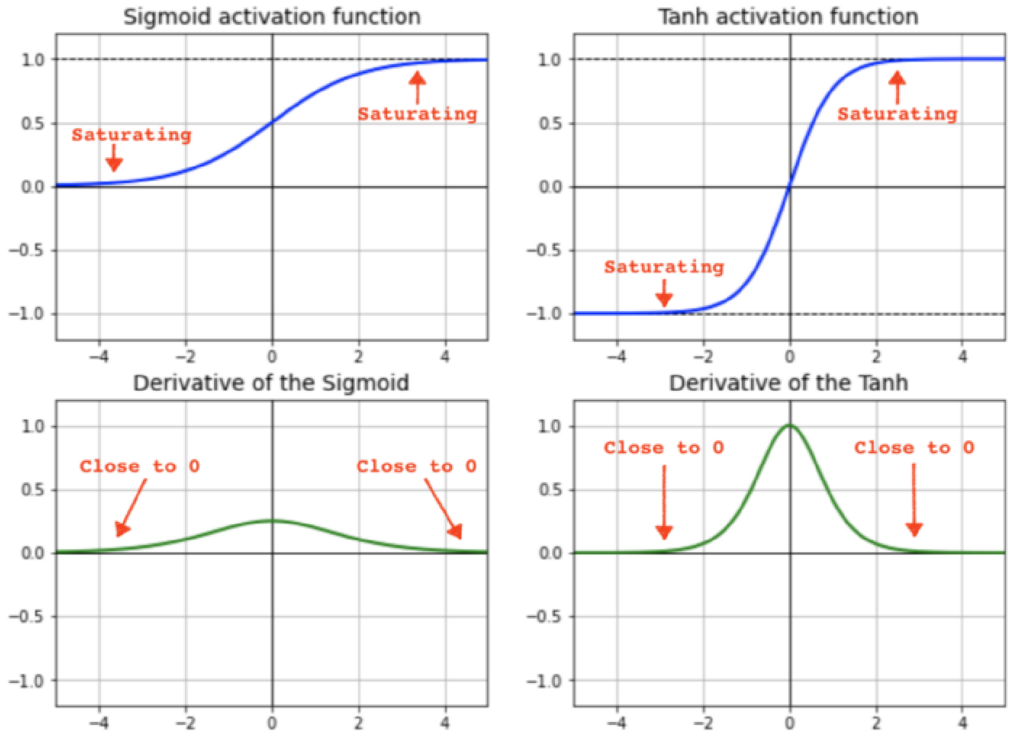
$$f^{[l]} = \left(W^{[l]}\right)^{\top}\boldsymbol{a}^{[l-1]}$$

Saturating Gradient Problem due to activation functions

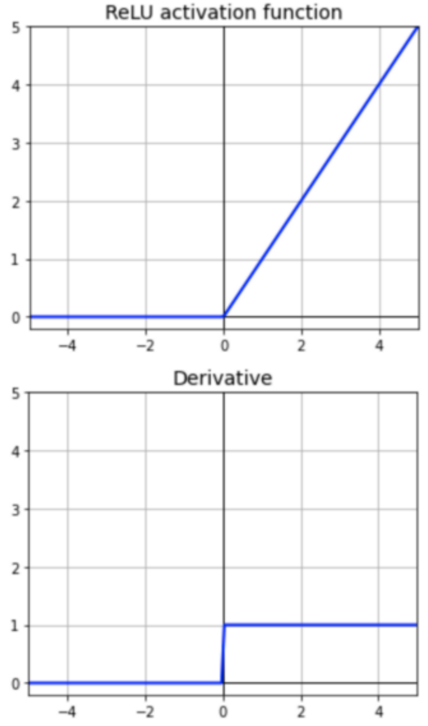
Mitigate with ReLU activation function

$g(f)$

$\dfrac{\partial g}{\partial f}$



Mitigation



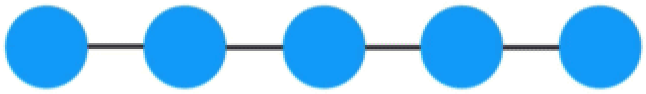
When x value far from 0, gradient → 0 (saturating)
When gradient ≈ 0, then weights don't update much

$$\Delta \boldsymbol{W} = \eta \nabla \varepsilon \approx 0$$

With ReLU, gradient is always 1 (for $x > 0$)
Can always update weights (for $x > 0$)

Image credit: https://towardsdatascience.com/why-rectified-linear-unit-relu-in-deep-learning-and-the-best-practice-to-use-it-with-tensorflow-e9880933b7ef

Vanishing Gradient Problem



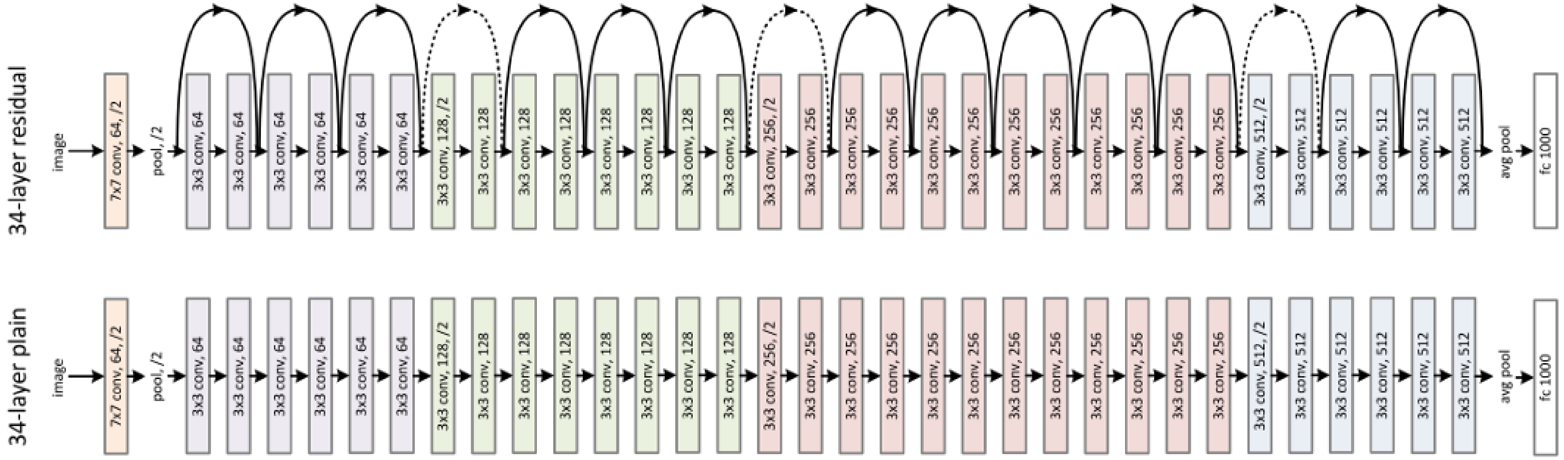
$$\hat{y}'\left(\boldsymbol{W}^{[1]}\right) = \frac{\partial g^{[L]}}{\partial \boldsymbol{W}^{[1]}} = \frac{\partial f^{[1]}}{\partial \boldsymbol{W}^{[1]}} \frac{\partial g^{[1]}}{\partial f^{[1]}} \cdots \frac{\partial g^{[l]}}{\partial f^{[l]}} \frac{\partial f^{[l+1]}}{\partial g^{[l]}} \frac{\partial g^{[l+1]}}{\partial f^{[l+1]}} \cdots \frac{\partial f^{[L]}}{\partial g^{[L-1]}} \frac{\partial g^{[L]}}{\partial f^{[L]}}$$

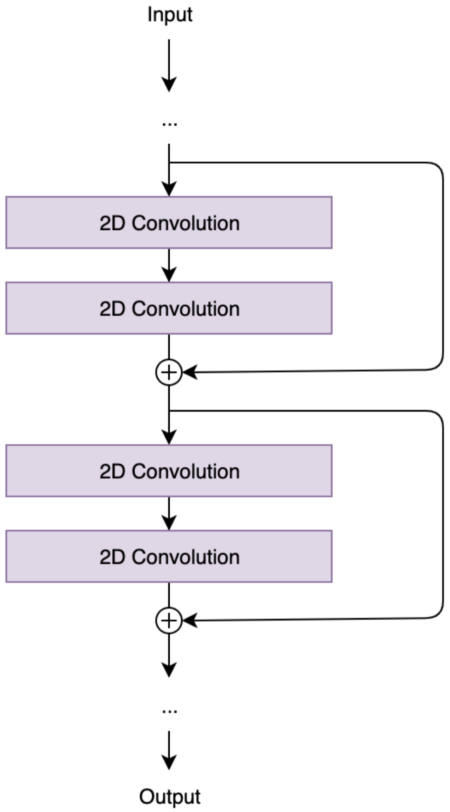
If some gradients are small (< 1),
multiplying many small numbers equals a very small number.
E.g., $0.5^{15} \approx 0.0003$

Image credit: https://towardsdatascience.com/understanding-rnns-lstms-and-grus-ed62eb584d90

Mitigating Vanishing Gradients in CNN:
Using architecture with “shortcut” connections

- ResNet (Residual Networks)
- Propagates residuals (forward) and gradients (backwards) through “**shortcut connections**”
- Gradients through shortcuts will not be as small



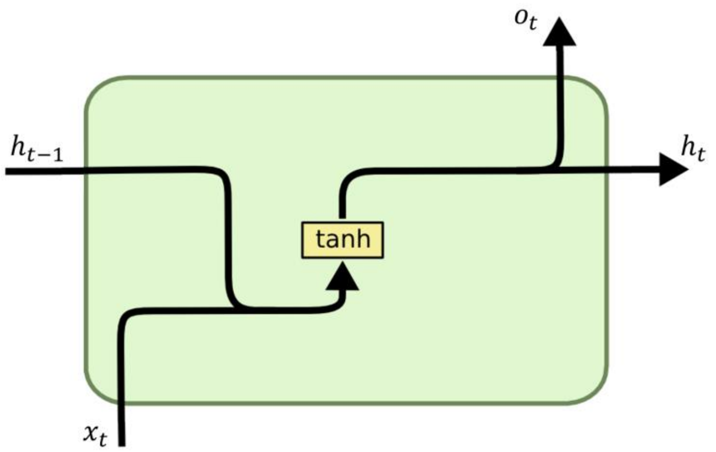


Further reading: https://towardsdatascience.com/vggnet-vs-resnet-924e9573ca5c
Image credit: https://www.kaggle.com/keras/resnet50

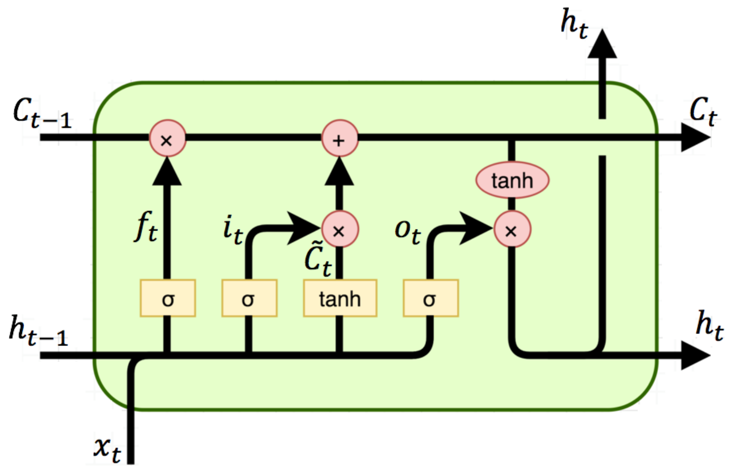
Advanced (details not in exam, but awareness is)

Mitigating Vanishing Gradients in RNN
Using architectures with "forget" gates

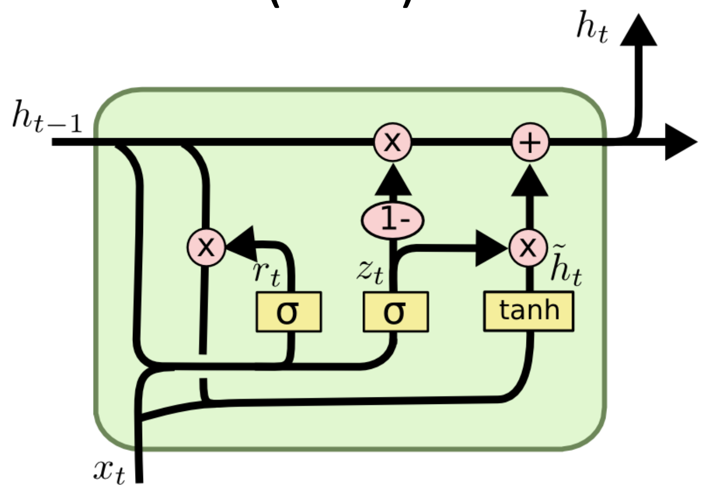
Plain RNN



Long-Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



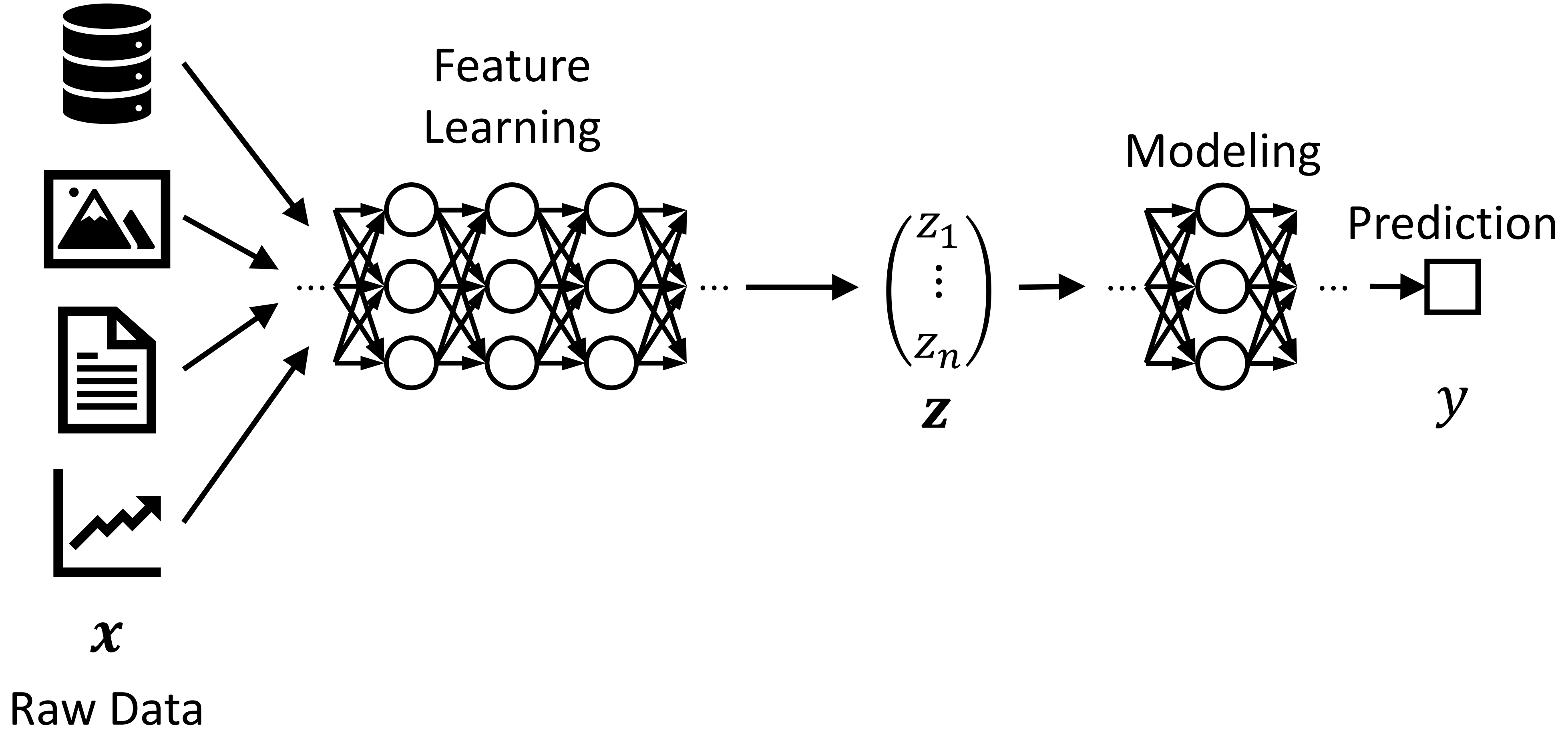
Includes **"forget" gates**

Image Credit: http://dprogrammer.org/rnn-lstm-gru
Further reading: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Wrapping Up

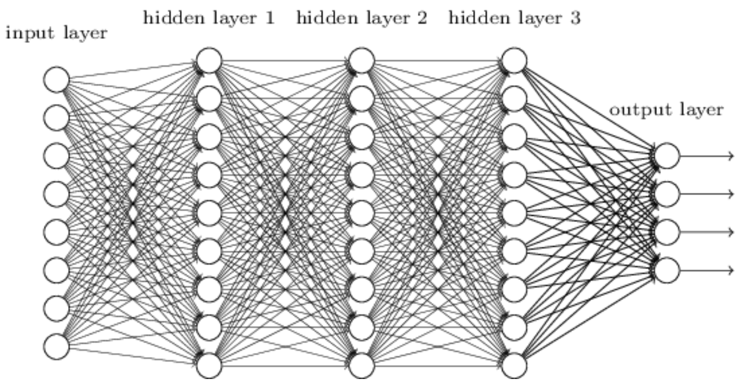
NUS National University of Singapore
Department of Computer Science
School of Computing

From Manual Feature Engineering To Architecture Engineering

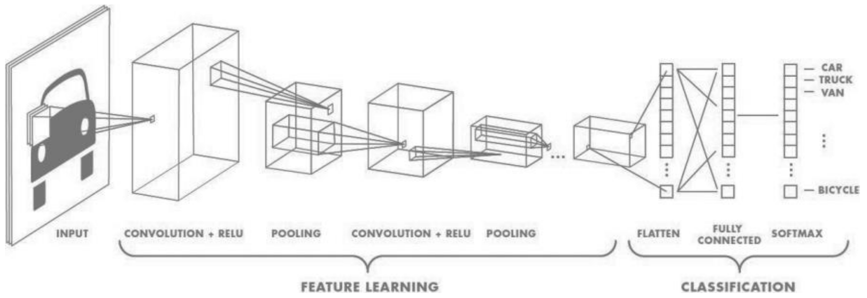


What did we learn?

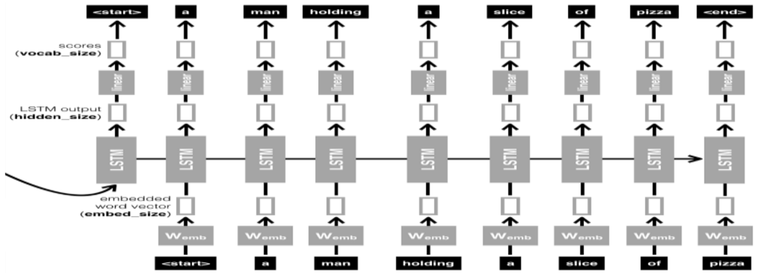
- Feature Engineering → Architecture Engineering



- **C**NN: exploits <u>spatial information</u> using **convolutions**



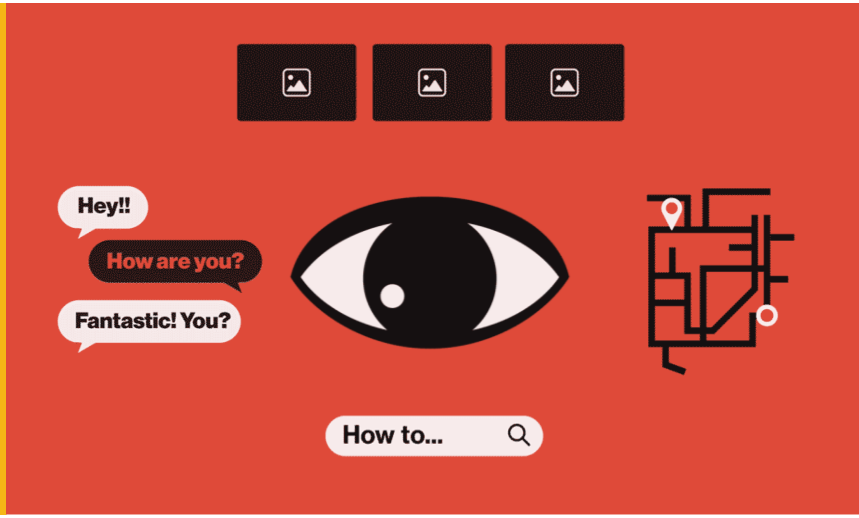
- **R**NN: exploits <u>history information</u> using **recurrence**



Grand issues with AI (Deep Learning)

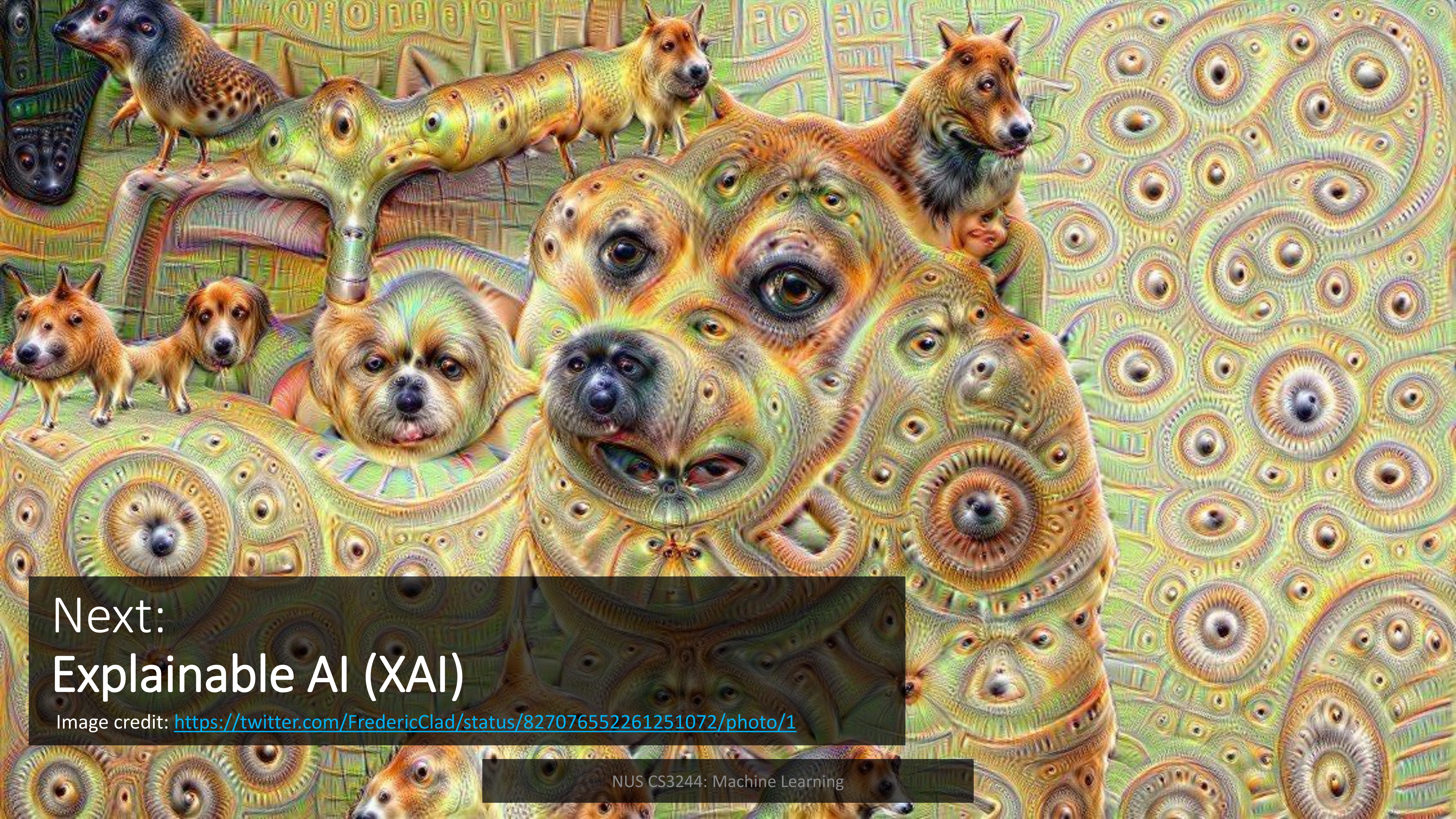
Lack of **Explainability**
[W11a]

Algorithmic Bias (Societal)
[W13a]



Data **Privacy**

Image credits:
https://miro.medium.com/max/2000/1*H4cW-_RCyHpu5FNtVaAPoQ.gif
https://www.insperity.com/wp-content/uploads/bias_1200x630.png
https://www.fightforprivacy.co/_nuxt/img/512f421.gif



Next:
Explainable AI (XAI)

Image credit: https://twitter.com/FredericClad/status/827076552261251072/photo/1