Deep Learning (DNN+CNN)

10 A

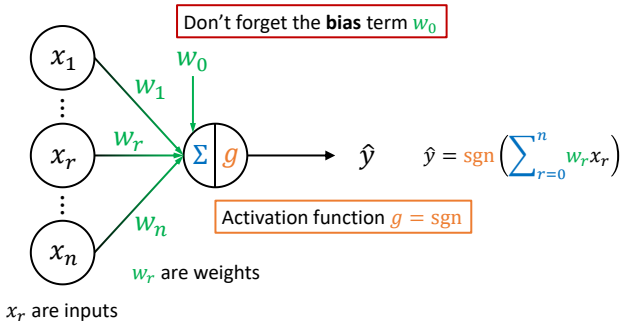
CS 3244 Machine Learning

NUS | Computing
National University of Singapore

Mid-Semester Anonymous Survey

- What worked
  - Slack interactions and Exercises (pre-lecture, in-class) are engaging.
  - Analogies (Min), Colored animations, Structured overviews (Brian) are effective.
  - Recaps (by TG leaders) are helpful and tutorial questions complement lectures.
- What to improve
  - Math. We will ensure to provide definitions of terms and expressions.
  - Too fast. I will slow down (but use more lecture slots).
  - Want more examples of how ML is used in real world. Will add to lectures.
  - Some tutorial questions (e.g., math proofs) disconnected from lectures. We will contextualize and align tutorial questions more closely with lectures.

Perceptron



Perceptron Learning Algorithm

1. Initialize weights $\boldsymbol{w}$
   - Could be all zero, or random small values
2. For each instance i with features $\boldsymbol{x}^{(i)}$
   - Classify $\hat{y}^{(i)} = \text{sgn}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)})$
3. Select one misclassified instance
   - Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(y - \hat{y})\boldsymbol{x}$
4. Iterate steps 2 to 3 until
   - Convergence (classification error < threshold), or
   - Maximum number of iterations

$$\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_r \\ \vdots \\ w_n \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_r \\ \vdots \\ w_n \end{pmatrix} + \eta(y - \hat{y}) \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_r \\ \vdots \\ x_n \end{pmatrix}$$

$$w_r \leftarrow w_r + \eta(y - \hat{y})x_r$$

Differentiable Activation Functions

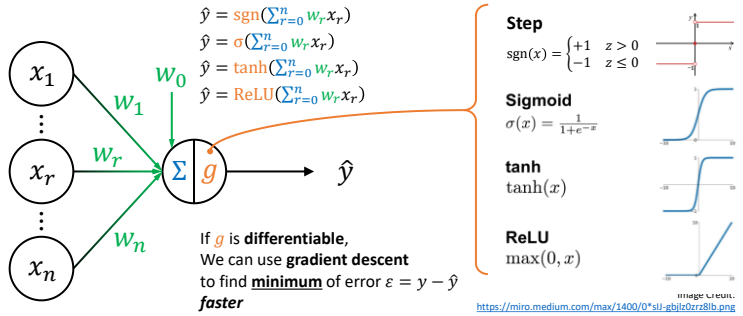
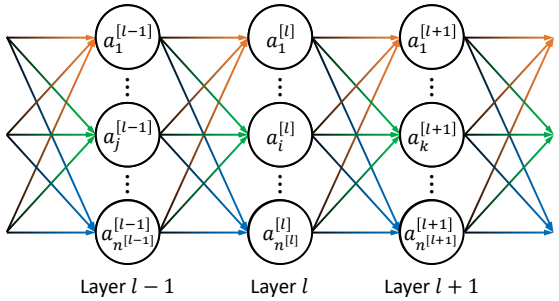


Image Credit: https://miro.medium.com/max/1400/0*sIJ-gbjlz0zrz8lb.png

Multi-Layer Perceptron (Neural Network)



Chain Rule

Consider composite function

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

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

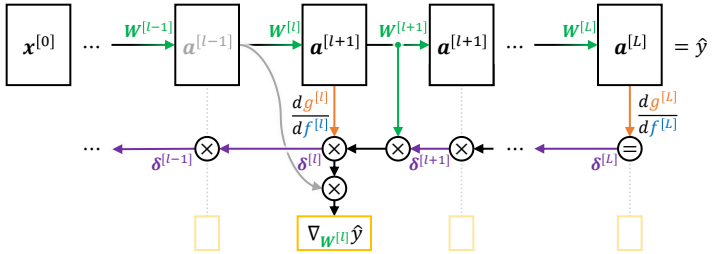
$$g'(x) = \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}$$

Intuition
Rate of change of g relative to x is the product of
- rates of change of g relative to f and
- rates of change of f relative to x

"If
- *a car travels 2x fast as a bicycle and*
- *the bicycle is 4x as fast as a walking man,*

then the car travels 2 × 4 = 8 times as fast as the man."
– George F. Simmons, Calculus with Analytic Geometry (1985)

Backward Propagation



Week 10A&B: 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

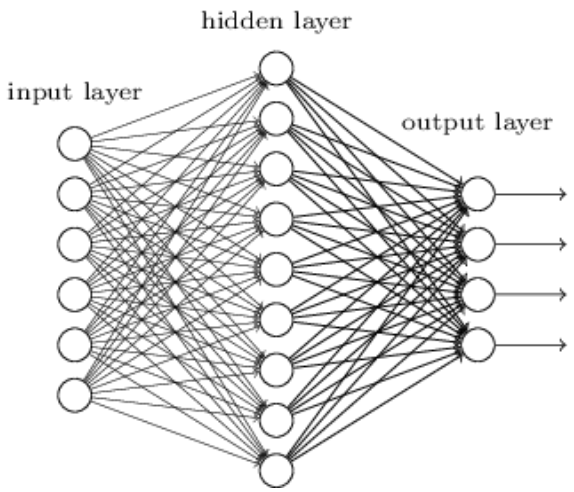
Week 10A: Lecture Outline

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

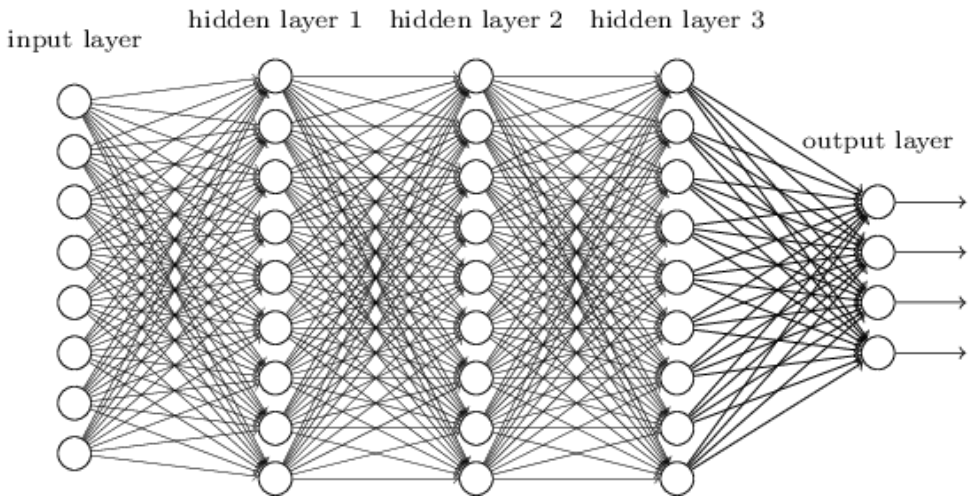
Deep Neural Network

Deep Neural Network = many hidden layers (≥3)

Shallow Network



Deep Network



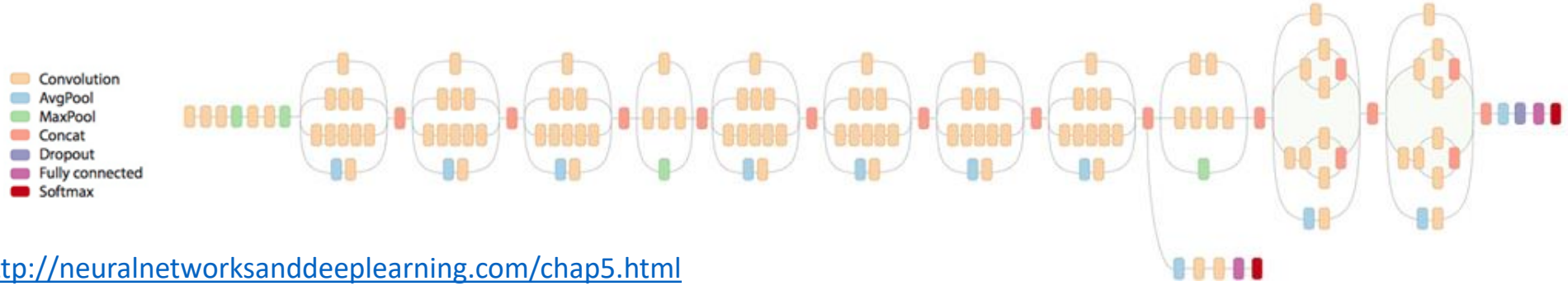
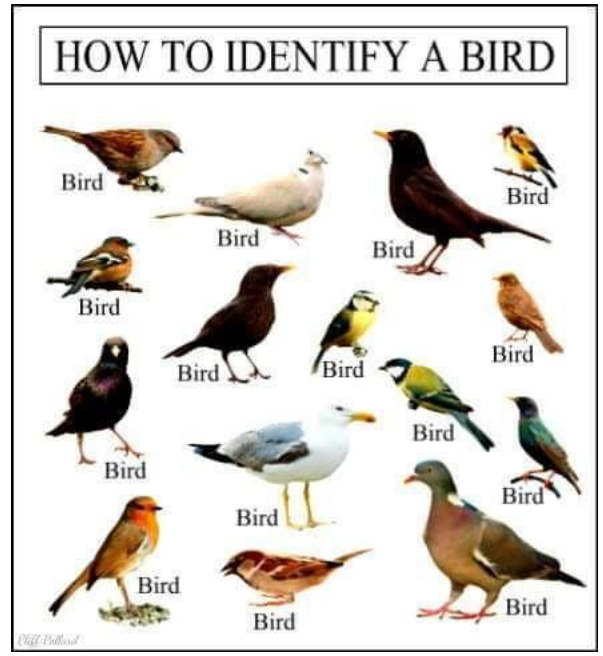


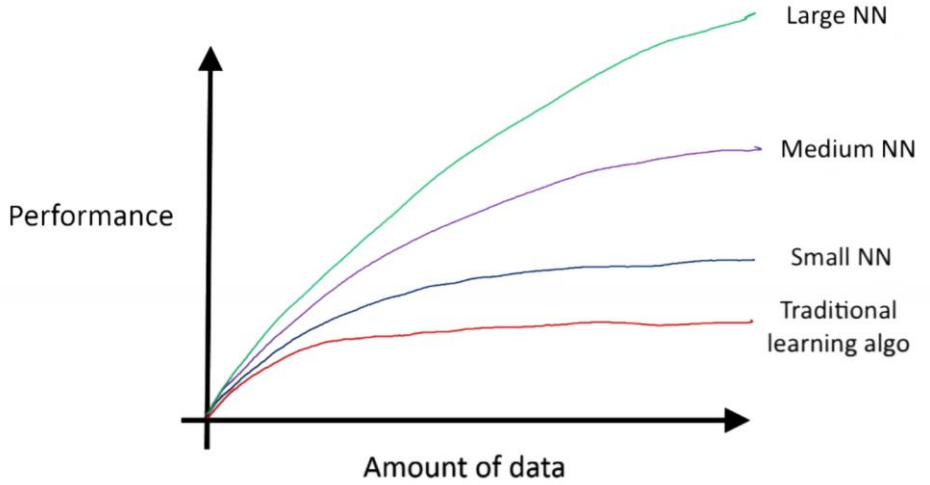
Image credit: http://neuralnetworksanddeeplearning.com/chap5.html
https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

Why Deep?

- Why need so many layers?
  - Need **many parameters**
  - Target functions of real-world tasks are **complex**
  - E.g., what is the function for recognizing birds or language?
- Why need so much training data?
  - Many parameters → Need more data
  - More data → Better performance
  - Avoid curse of dimensionality

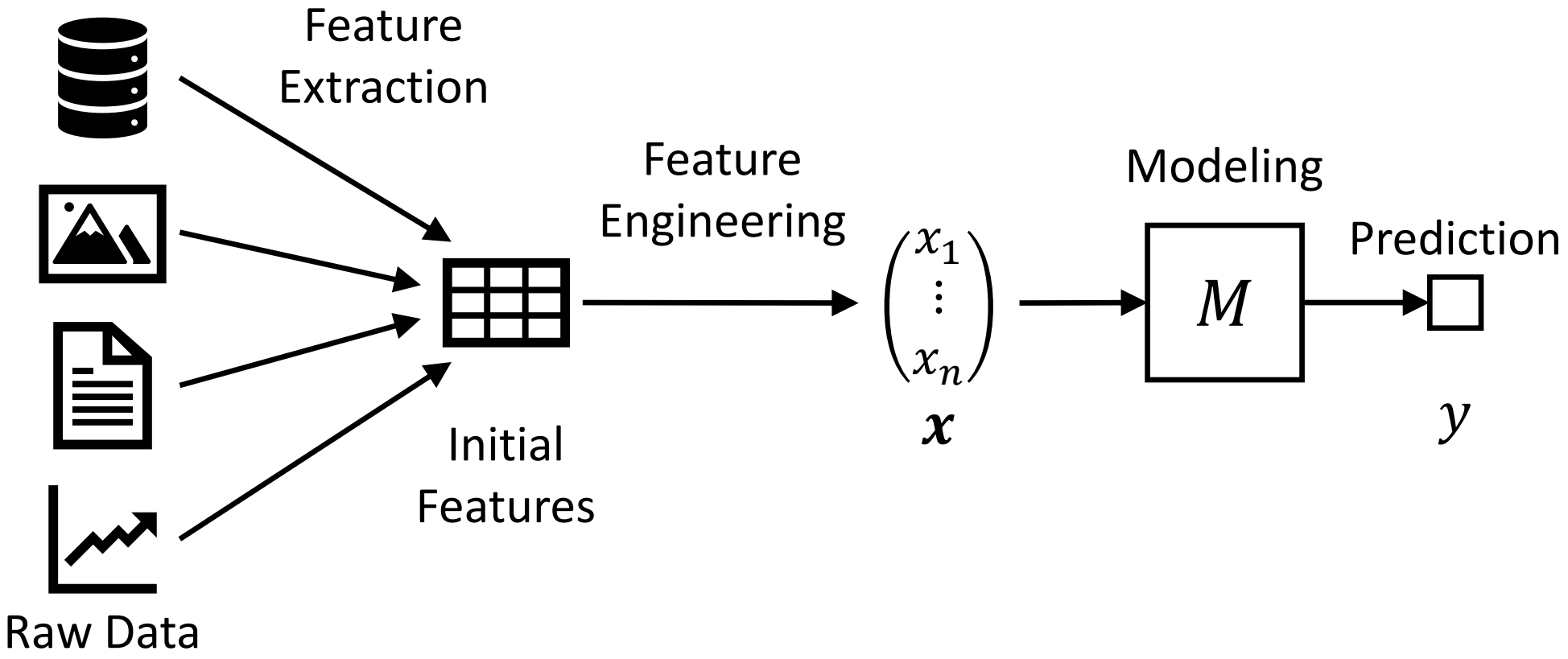


https://9gag.com/gag/ax9Roon

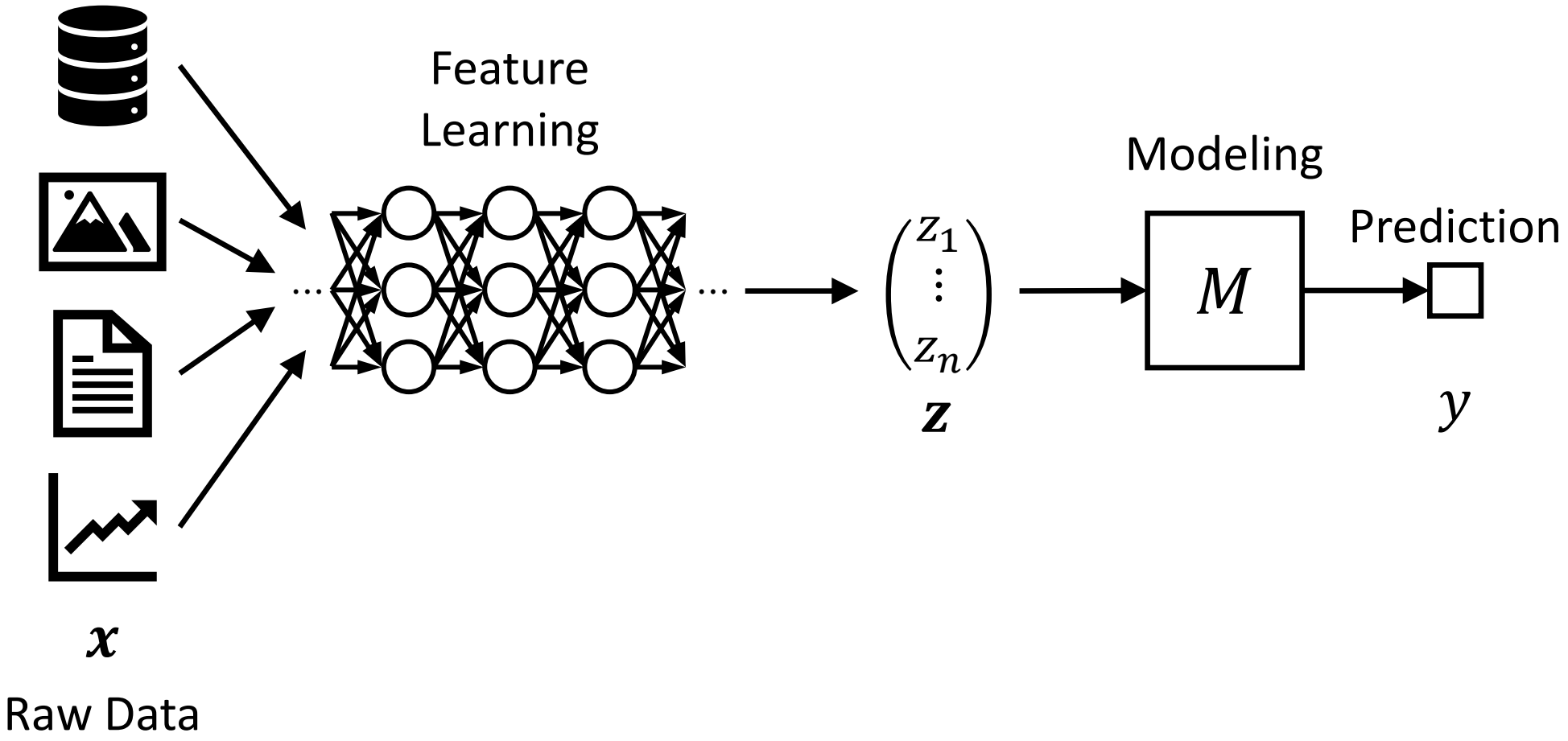


Andrew Ng https://youtu.be/LcfLo7YP8O4

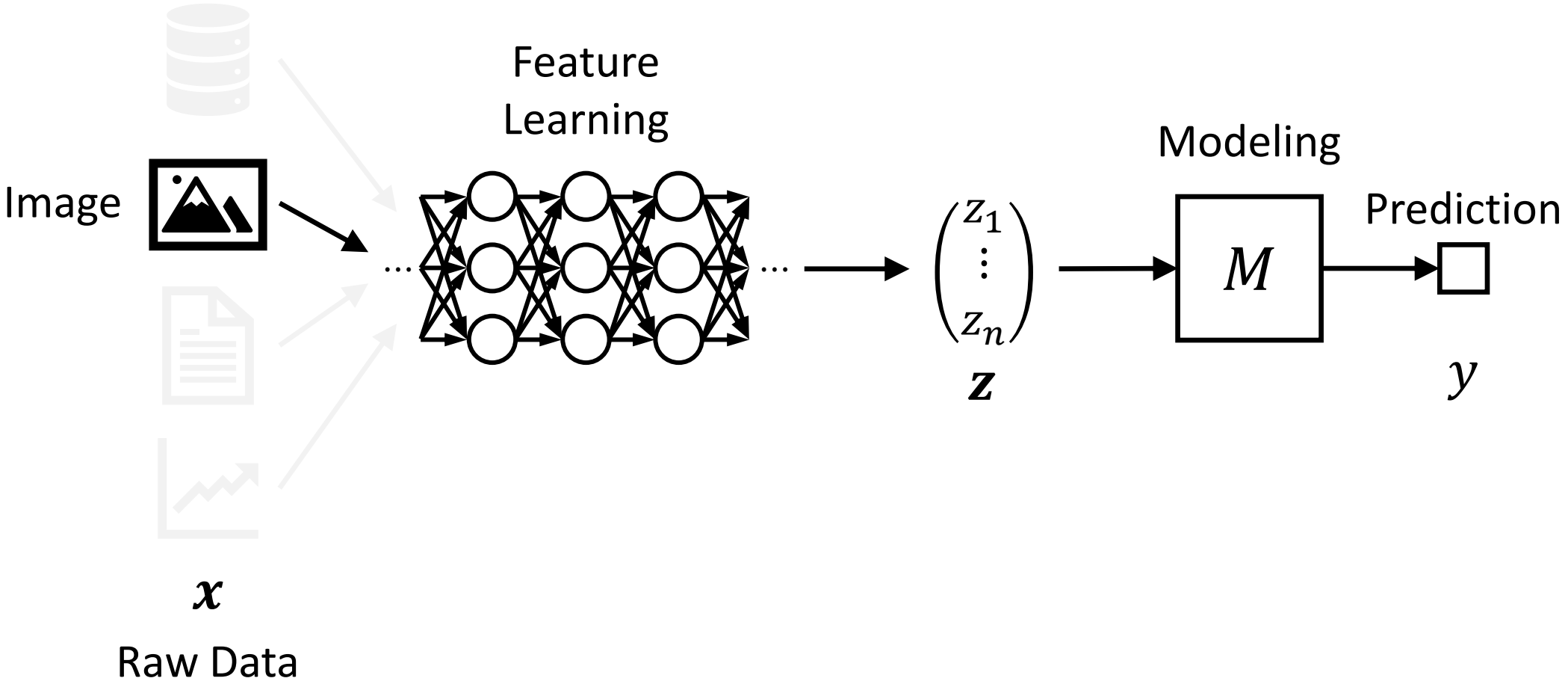
Feature Extraction/Engineering → Modeling



From Manual Feature Engineering To Automatic Feature Learning



From Manual Feature Engineering To Automatic Feature Learning



Convolutional Neural Networks (CNN)

Applications of CNN

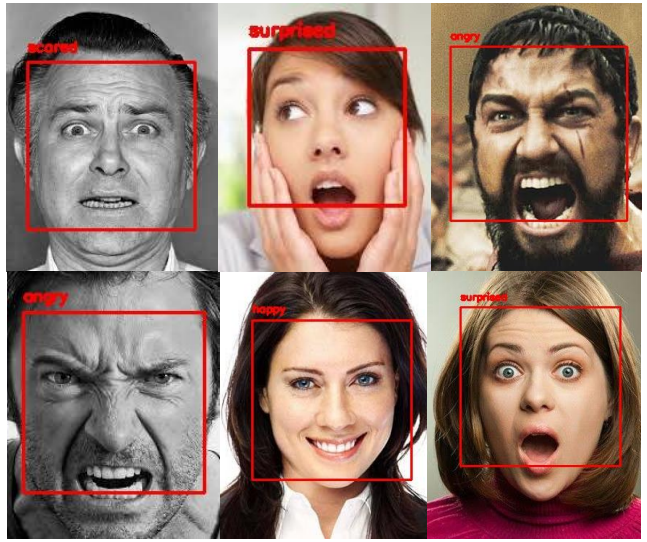
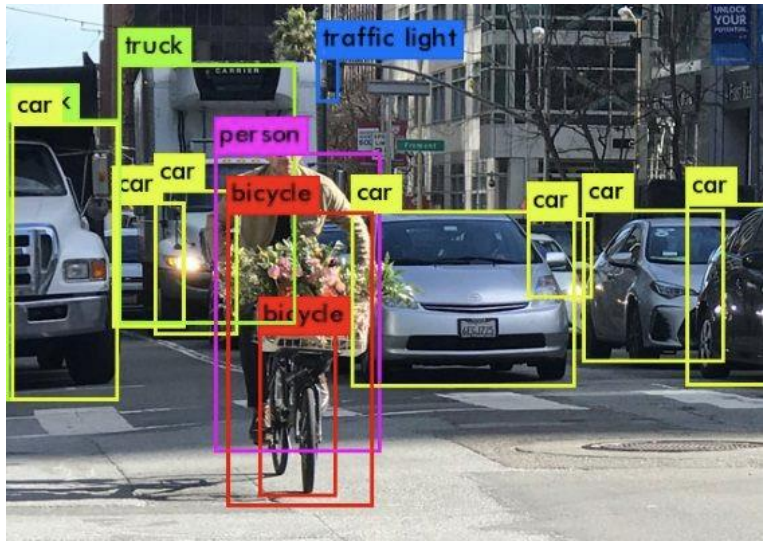


Image Classification
e.g., face emotions



Object Detection
e.g., self-driving cars

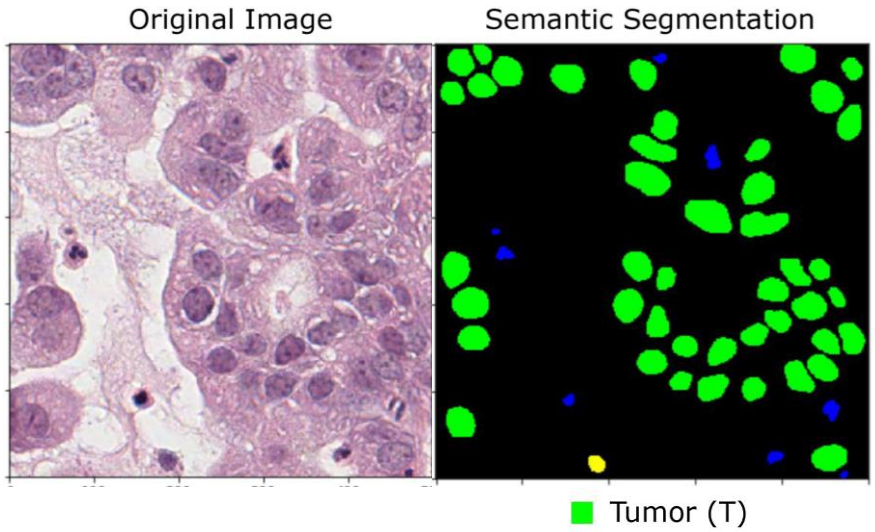


Image Segmentation
e.g., cancer cell detection

Image credit:
https://monica-dommaraju.medium.com/analysis-of-deep-learning-based-object-detection-f14d5138148
https://ajp.amjpathol.org/article/S0002-9440(18)31121-0/fulltext
https://appliedmachinelearning.blog/2018/11/28/demonstration-of-facial-emotion-recognition-on-real-time-video-using-cnn-python-keras/

FOODAi Demo

https://foodai.org

Try out our demo below or visit our developer portal for our API services.

To try our demo, you can click the upload icon to choose the image, or copy and paste the image or drag and drop the image from desktop or internet to the upload area.

Chicken rice

Boiled kampung chicken

Chicken porridge

Fish porridge

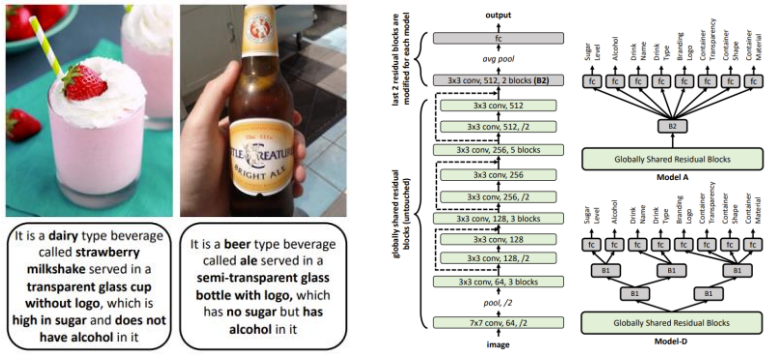
Fried fish porridge

Register for FoodAI API Free Trial

tastehealthy

Backend Models

Food & Drink Recognition



Food Recommendation

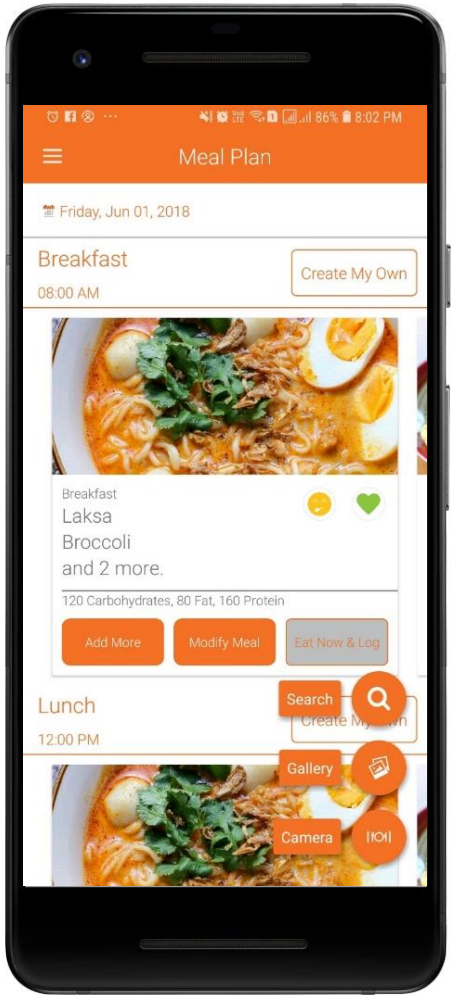
RecGAN

Generator

$$\mathbf{x}_t^u = RELU(\mathbf{W}_{xh}^u \mathbf{h}_{t-1}^u + \mathbf{W}_{xk}^u \mathbf{y}_t^u)$$
$$\mathbf{r}_t^u = \sigma(\mathbf{W}_{rh}^u \mathbf{h}_{t-1}^u + \mathbf{W}_{rk}^u \mathbf{y}_t^u)$$
$$\mathbf{m}_t^u = tanh(\mathbf{W}_h(\mathbf{r}_t^u \cdot \mathbf{h}_{t-1}^u) + \mathbf{W}_{input} \mathbf{y}_t^u)$$
$$\mathbf{h}_t^u = (1 - \mathbf{x}_t^u) \cdot \mathbf{h}_{t-1}^u + \mathbf{x}_t^u \cdot \mathbf{m}_t^u$$

Discriminator

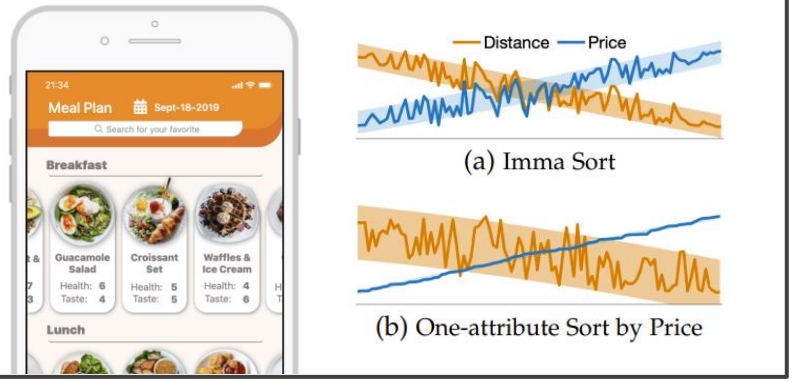
$$\hat{\mathbf{x}}_t^u = RELU(\mathbf{V}_{xh}^u \hat{\mathbf{h}}_{t-1}^u + \mathbf{V}_{xk}^u \mathbf{y}_t^u)$$
$$\hat{\mathbf{r}}_t^u = \sigma(\mathbf{V}_{rh}^u \hat{\mathbf{h}}_{t-1}^u + \mathbf{V}_{rk}^u \mathbf{y}_t^u)$$
$$\hat{\mathbf{m}}_t^u = tanh(\mathbf{V}_h(\hat{\mathbf{r}}_t^u \cdot \hat{\mathbf{h}}_{t-1}^u) + \mathbf{V}_{input} \mathbf{y}_t^u)$$
$$\hat{\mathbf{h}}_t^u = (1 - \hat{\mathbf{x}}_t^u) \cdot \hat{\mathbf{h}}_{t-1}^u + \hat{\mathbf{x}}_t^u \cdot \hat{\mathbf{m}}_t^u$$



Frontend UI/UX

Family Food Logging

Multi-Attribute Sorting



Lyu, Y., Gao, F., Wu, I. S., & Lim, B. Y. 2020. Imma Sort by Two or More Attributes With Interpretable Monotonic Multi-Attribute Sorting. TVCG.
Park, H., Bharadhwaj, H., and Lim, B. Y. 2019. Hierarchical Multi-Task Learning for Healthy Drink Classification. *IJCNN.*
Bharadhwaj, H., Park, H., Lim, B. Y.. 2018. RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems. *RecSys '18.*
Lukoff, K., Li, T., Zhuang, Y., & Lim, B. Y. 2018. TableChat: Mobile Food Journaling to Facilitate Family Support for Healthy Eating. *CSCW '18.*

NUS Ubicomp Lab
Apps and Analytics for Smart Cities and Health

Images as 2D matrices

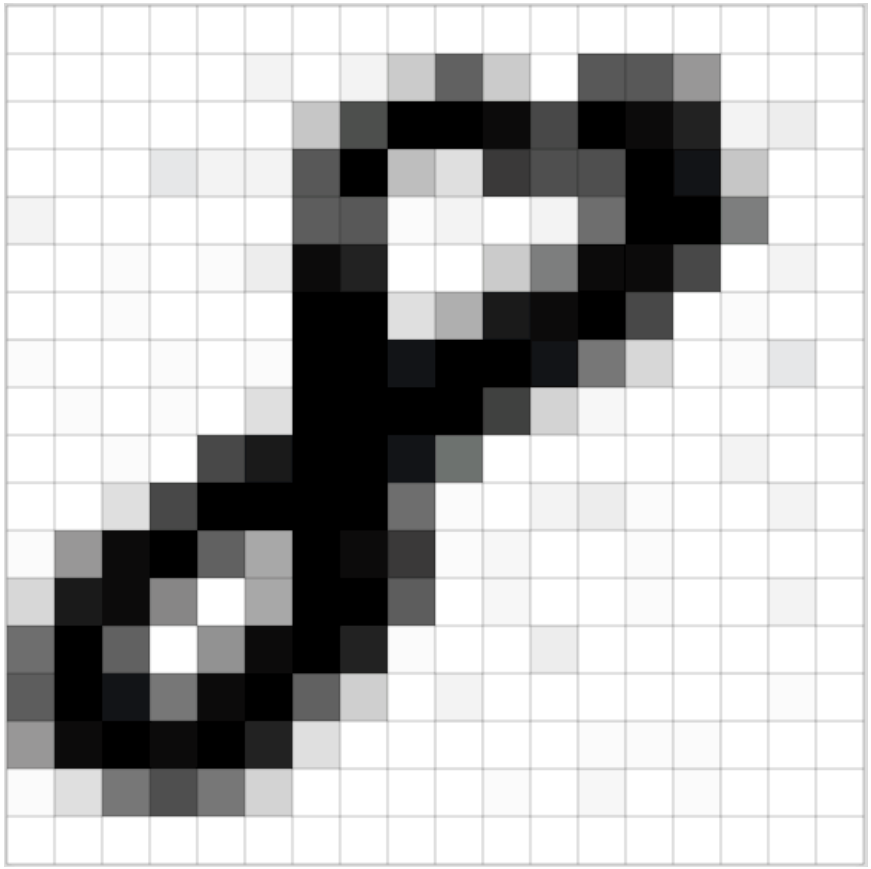
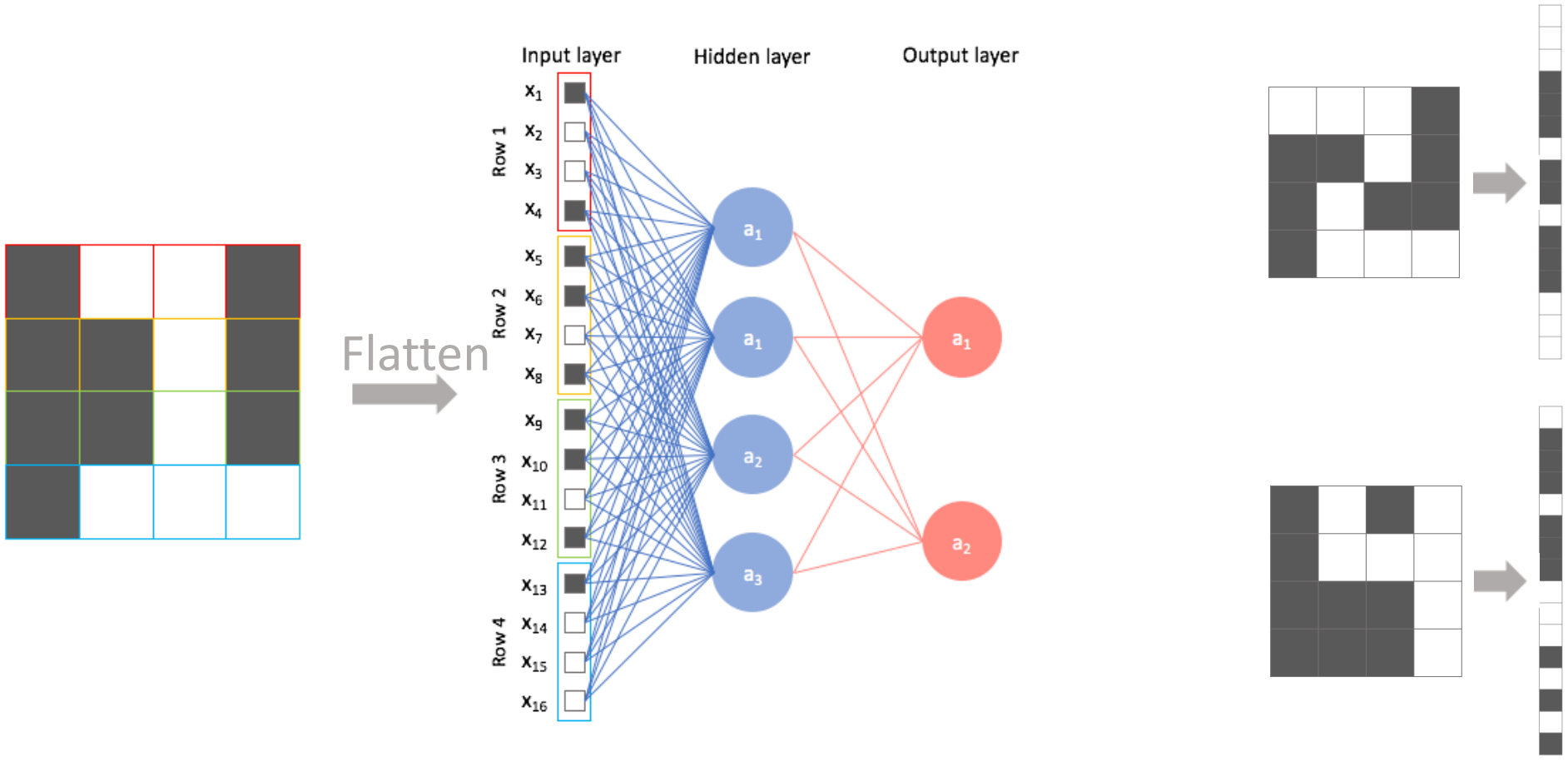
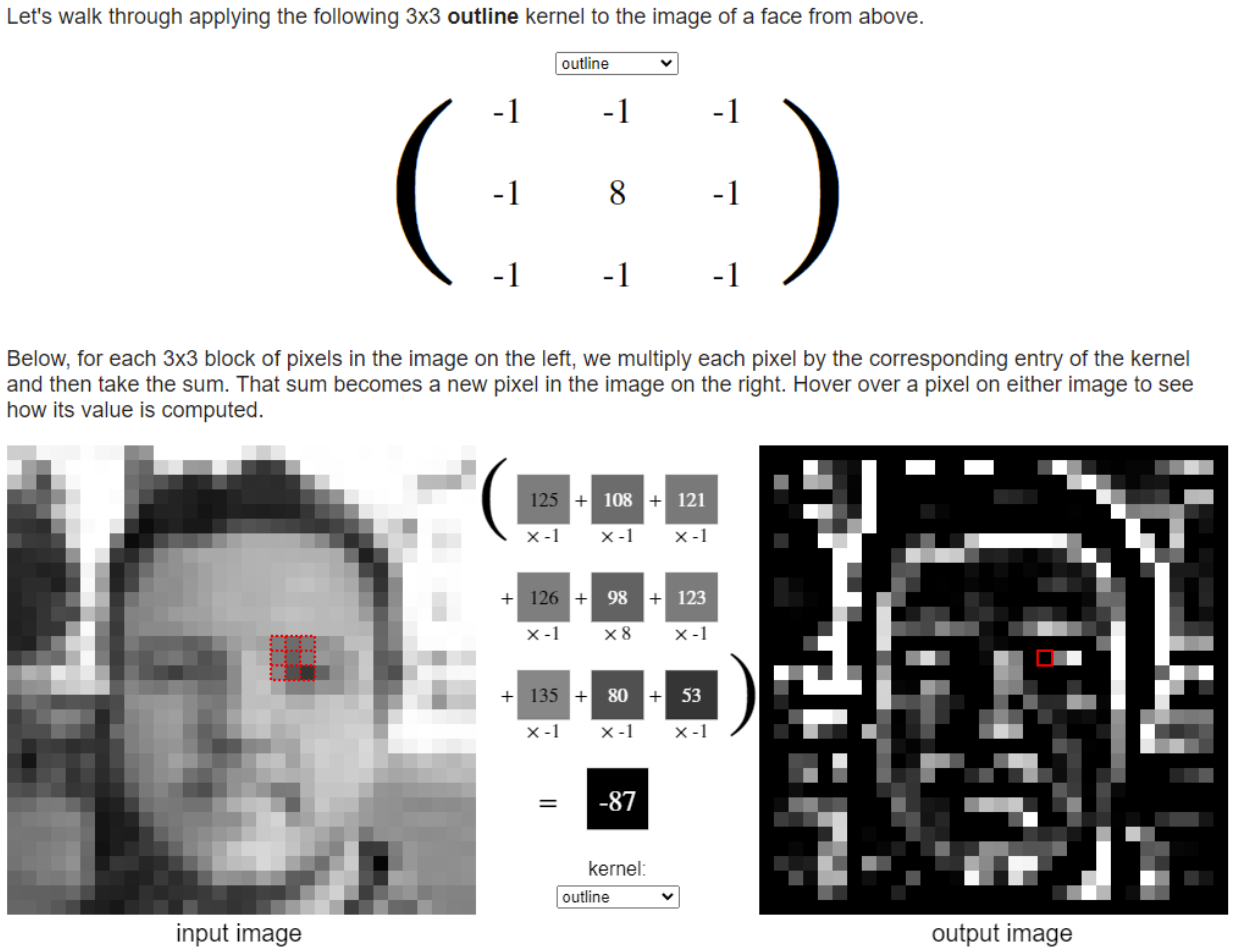


Image credit

Image Feature Extraction with
Fully Connected Neural Networks (Multi-Layer Perceptron)



Reduce parameters for images:
Exploit **Spatial** Relations with **Convolutions**



Manually finding good filters is tedious

Further study:
https://setosa.io/ev/image-kernels/

Insert Web Page

This app allows you to insert secure web pages starting with https:// into the slide deck. Non-secure web pages are not supported for security reasons.

Please enter the URL below.

https:// setosa.io/ev/image-kernels/

Note: Many popular websites allow secure access. Please click on the preview button to ensure the web page is accessible.

Preview

Learning Convolutional Filters

High-level Feature Detection

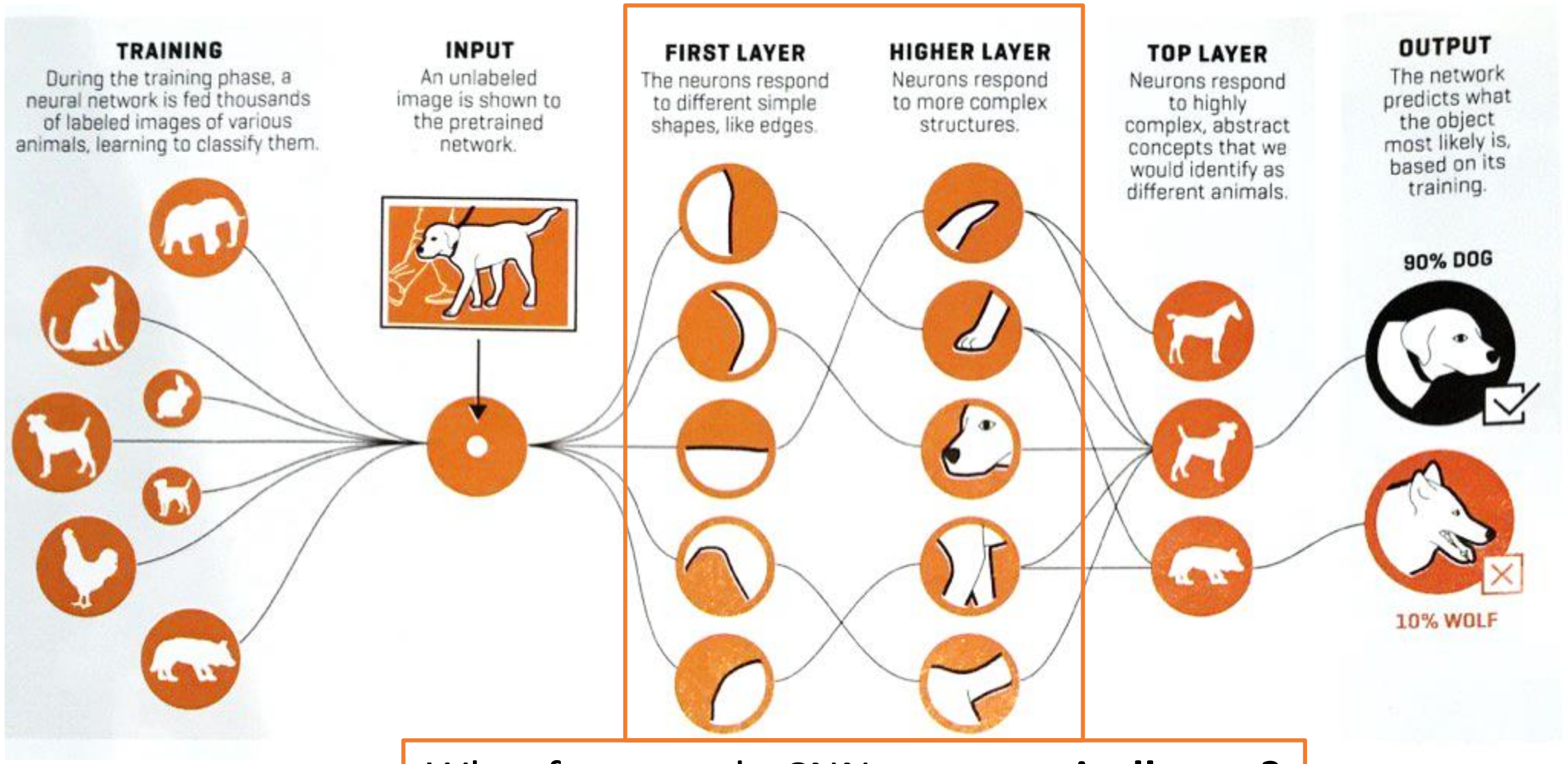
Eyes, Nose, Mouth
Facial Hair

Wheels, Headlights,
Bonnet/Hood

Fish, Rice,
Vegetables

What features do CNNs **automatically** use?

Feature Detectors: Intuition of Neuron Kernels in Layers



What features do CNNs **automatically** use?

W10 Pre-Lecture Task (due before next Mon)

Watch

- Who Invented A.I.? - The Pioneers of Our Future by ColdFusion

Play

- https://distill.pub/2018/building-blocks/
  - Don't worry about reading the whole article

Discuss

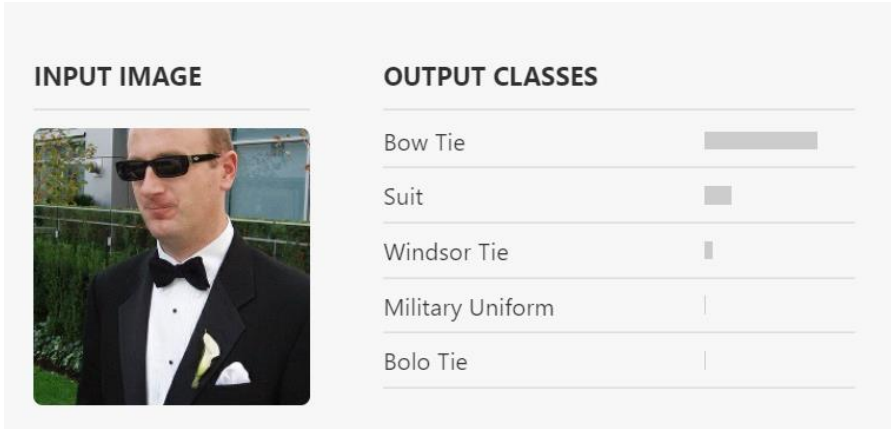
1. Identify what is strange, funny, or erroneous in the deep learning model in Building-Blocks
2. Take a screenshot of the issue and share with your tutorial mates
3. Try to explain why the model was behaving as identified
3. Post a 2–3 sentence description to the topic in your tutorial group: #tg-xx

Pre-Lecture Activity for W10

The deep learning model would have a bit of a problem when it is trying to identify **multiple objects** in the same image, and in some cases, it would not detect the other object/class.

In this picture, it can be seen that the **dog** is well-identified, … the **kitten**, however, was not detected at all. …

Need to understand prediction task of model.
Multi-Class vs. Multi-Label



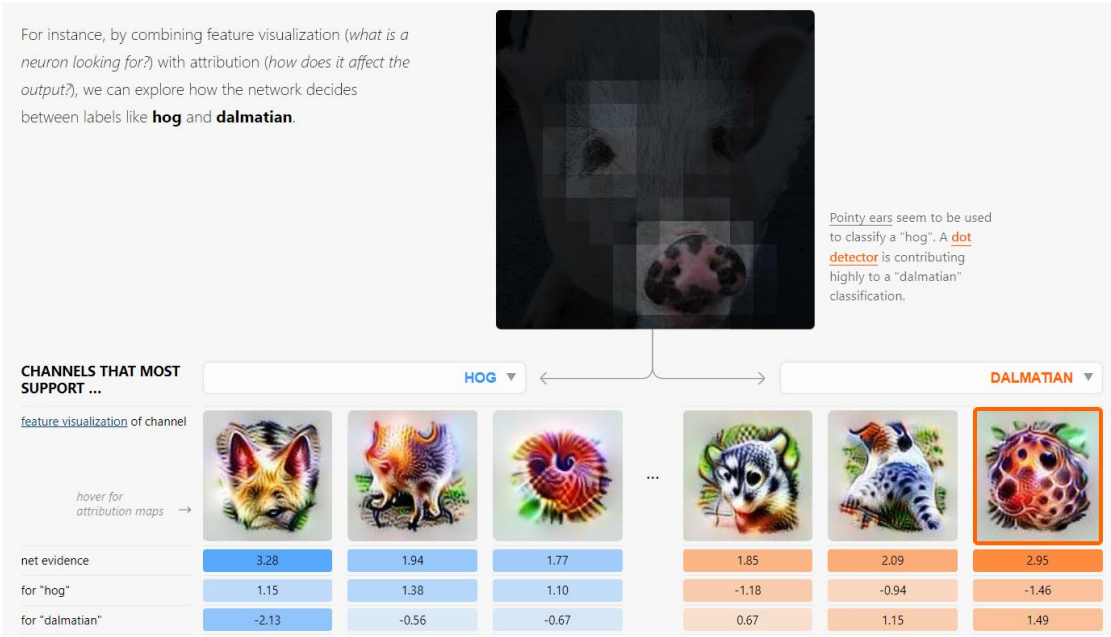
In this picture, if you ask **me (a human) to identify** the picture, I would say "**groom**". However, the output classes do not even list out "groom" in the output classes. Instead, it list out **bow tie** and **suit** as the top 2 output classes, which are the entities that the groom is wearing.

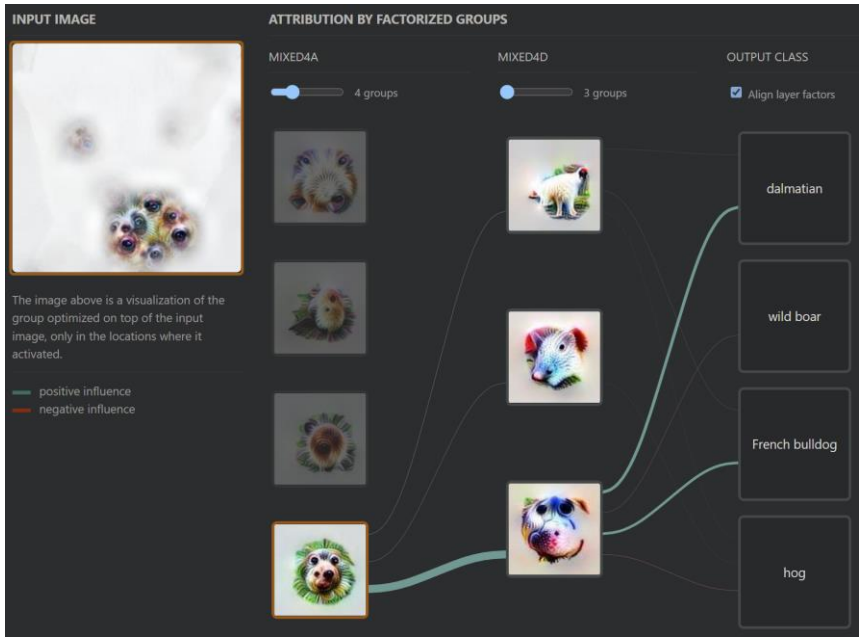
Need to understand scope of prediction.
Finite class labels vs. Open-ended

My guess is that the model looks at a localised part of the image to make a prediction of the object. So things like bow ties and suits which are quite simplistic in its outline are easily identified and carry more weight. However, to identify the groom, the model may need to see the

Can use Explainable AI (XAI)
To try to understand with less guessing

CNNs predict using *Weighted* **Feature Kernels**





When looking at the image of the pig, I thought it was interesting that

- the classifier had strong evidence to think that the image was a **dalmatian** due to the **presence of spots**.
- But ultimately it decided that the image was one of a **hog** due to the **pointy ears** which it **<u>weighted more</u>** than the spots.

The neuron seems to behave like this as that portion has **<u>many black spots</u>** which aligns more with

- **black snouts** commonly found in French bulldogs
- **black spots** in dalmatians.

Analogy: activations of different filters learned by CNNs is like seeing the image through different lens filters

How to **automatically** learn these features?

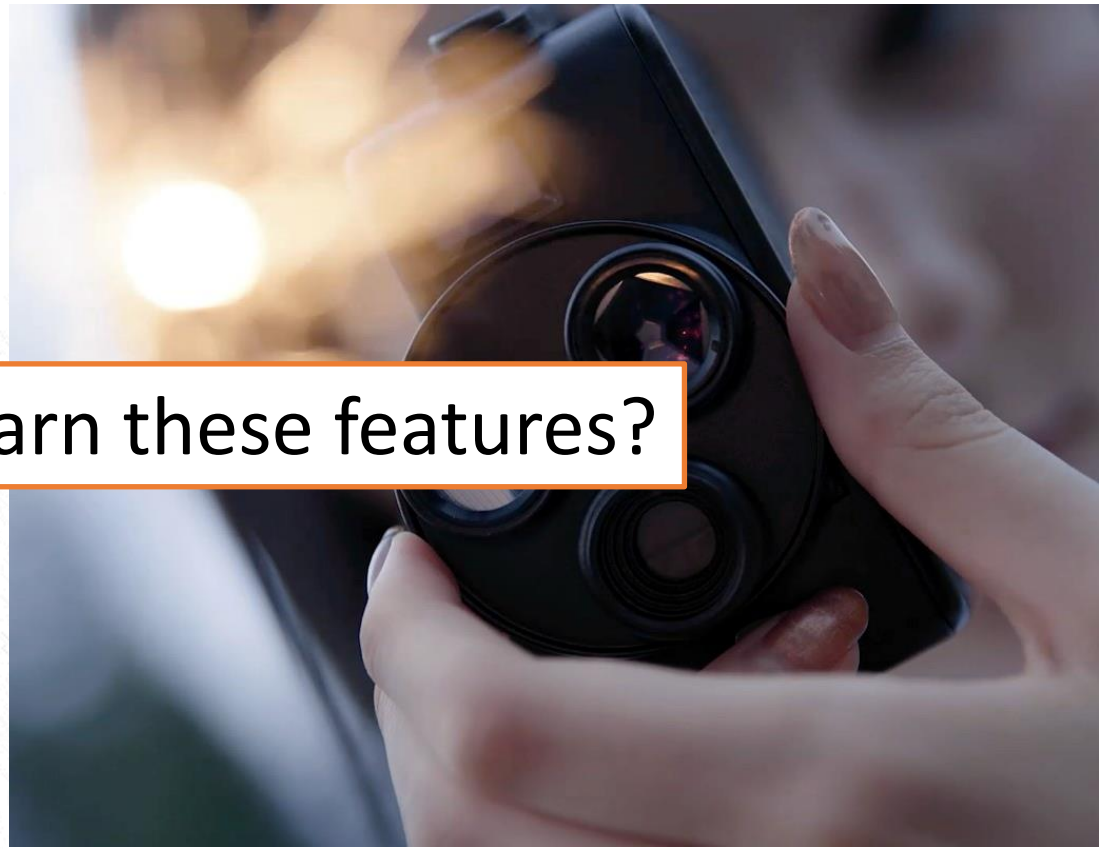


Image credit: https://www.amazon.com/Godefa-Samsung-Andriod-Smartphone-Universal/dp/B07RQRLQYH
https://www.yankodesign.com/2020/02/17/this-retro-inspired-camera-records-dreamy-looking-gifs-that-replicate-vintage-8mm-film/

Analogy: kernel update ≡ glass lens grinding

Image Credit: https://www.schneider-om.com/precision-optics/processes/polishing.html

Convolutions: Kernel Size, Stride, Padding

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

Stride $s \neq 1$

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

Padding $p \neq 0$

$$\boldsymbol{W} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} & & & & \\ & 3 & 3 & 4 & \\ & 3 & 3 & 5 & \\ & 3 & 3 & 4 & \\ & & & & \end{pmatrix} \quad \boldsymbol{W} * \boldsymbol{x} = \begin{pmatrix} 0+3+3 & 0-3-3 \\ 3+3+0 & -3-3+0 \end{pmatrix}$$

What are the Kernel Size, Stride, Padding?

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

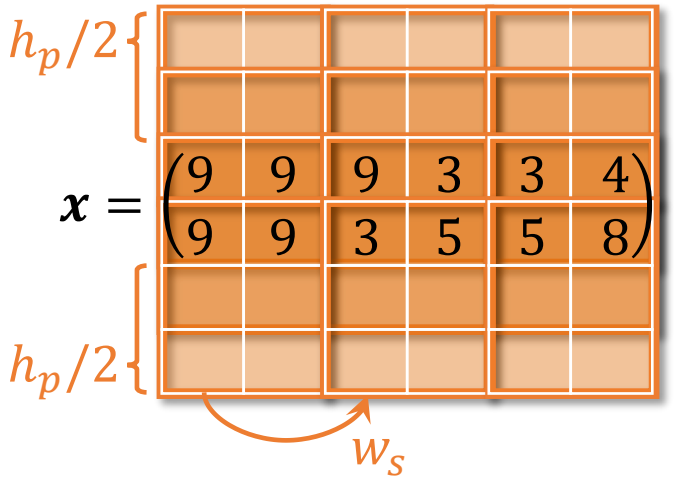
In Slack #lecture

1. **Write** answer to thread
    1. Kernel Size = ?
    2. Stride = ?
    3. Padding = ?
2. **Emote** (👍 :+1:) to vote for answer

What are the Kernel Size, Stride, Padding?

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

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



$$\boldsymbol{x} = \begin{pmatrix} 9 & 9 & 9 & 3 & 3 & 4 \\ 9 & 9 & 3 & 5 & 5 & 8 \end{pmatrix}$$

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

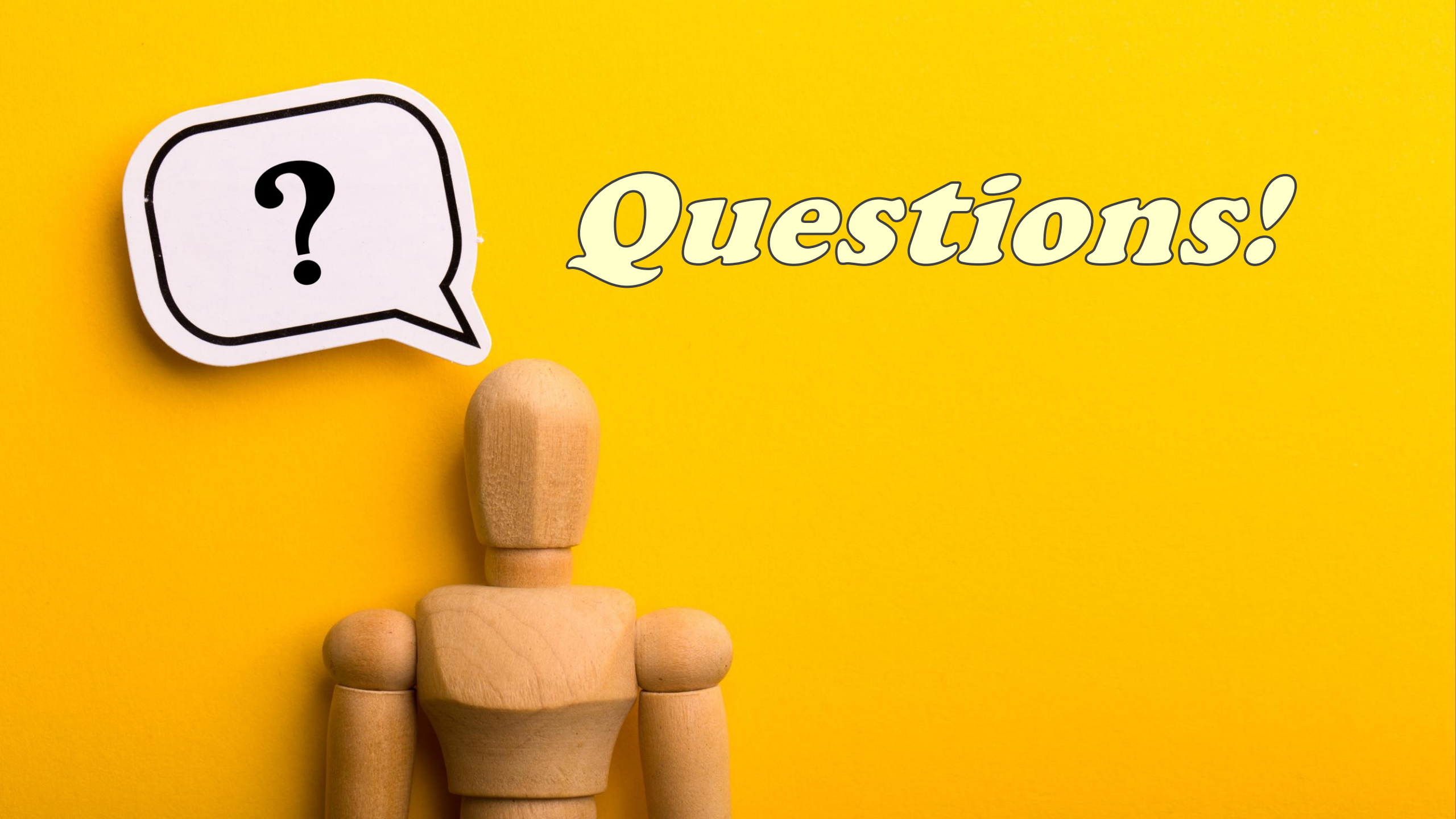
Hyperparameters

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

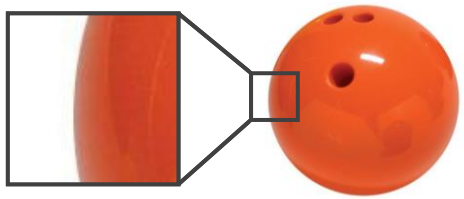
Chosen manually, or automatically with hyperparameter tuning

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

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


?
Questions!

Multi-Channel Convolutions



$$\boldsymbol{W}_{11} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{X}_1 = \begin{pmatrix} 9 & 9 & 3 & 3 & 4 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 3 & 3 & 5 & 5 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 9 & 3 & 3 & 4 \end{pmatrix} \quad \boldsymbol{W}_{11} * \boldsymbol{X}_1 = \begin{pmatrix} -6-6-6 & -6+1+2 & 1+2+2 \\ -6-6-6 & 1+2+1 & 2+2+2 \\ -6-6-6 & 2+1-6 & 2+2+1 \end{pmatrix} = \begin{pmatrix} -18 & -3 & 5 \\ -18 & 4 & 6 \\ -18 & -3 & 5 \end{pmatrix}$$

$$\boldsymbol{W}_{12} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{X}_2 = \begin{pmatrix} 9 & 9 & 1 & 1 & 2 \\ 9 & 1 & 1 & 2 & 3 \\ 9 & 1 & 1 & 3 & 3 \\ 9 & 1 & 1 & 2 & 3 \\ 9 & 9 & 1 & 1 & 2 \end{pmatrix} \quad \boldsymbol{W}_{12} * \boldsymbol{X}_2 = \begin{pmatrix} -8-8-8 & -8+1+2 & 1+2+2 \\ -8-8-8 & 1+2+1 & 2+2+2 \\ -8-8-8 & 2+1-8 & 2+2+1 \end{pmatrix} = \begin{pmatrix} -24 & -5 & 5 \\ -24 & 4 & 6 \\ -24 & -5 & 5 \end{pmatrix}$$

$$\boldsymbol{W}_{13} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{X}_3 = \begin{pmatrix} 9 & 9 & \blacksquare & \blacksquare & \blacksquare \\ 9 & \blacksquare & \blacksquare & \blacksquare & 1 \\ 9 & \blacksquare & \blacksquare & \blacksquare & 1 \\ 9 & \blacksquare & \blacksquare & \blacksquare & 1 \\ 9 & 9 & \blacksquare & \blacksquare & \blacksquare \end{pmatrix} \quad \boldsymbol{W}_{13} * \boldsymbol{X}_3 = \begin{pmatrix} -9-9-9 & -9+1+2 & 0+1+1 \\ -9-9-9 & 1+2+1 & 1+1+1 \\ -9-9-9 & 2+1-9 & 1+1+0 \end{pmatrix} = \begin{pmatrix} -27 & -6 & 2 \\ -27 & 4 & 3 \\ -27 & -6 & 2 \end{pmatrix}$$

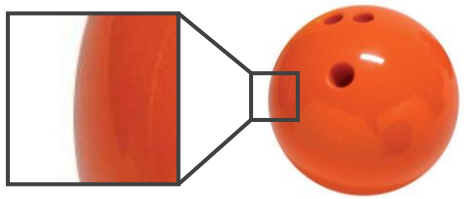
$$+ \rightarrow \sum_{r=1}^{c=3} \boldsymbol{W}_{1r} * \boldsymbol{X}_r = \begin{pmatrix} -69 & -14 & 12 \\ -69 & 12 & 15 \\ -69 & -14 & 12 \end{pmatrix}$$

Multi-channel input
One-channel output

Kernels (Filters)

Input (Image)

Multi-Channel Convolutions (another example)



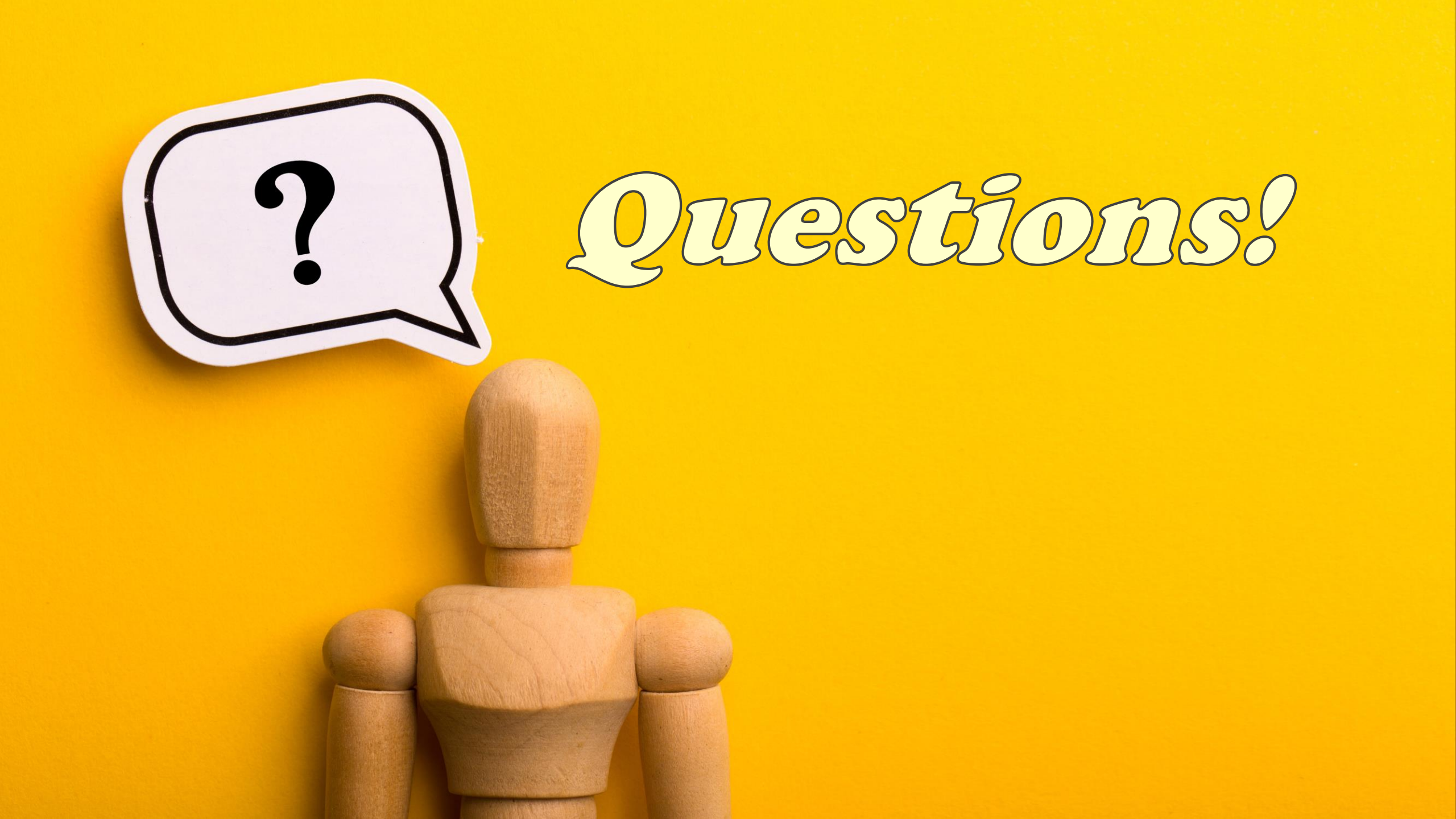
$$\boldsymbol{W}_{21} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{X}_1 = \begin{pmatrix} 9 & 9 & 3 & 3 & 4 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 3 & 3 & 5 & 5 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 9 & 3 & 3 & 4 \end{pmatrix} \quad \boldsymbol{W}_{21} * \boldsymbol{X}_1 = \begin{pmatrix} -6-6-6 & -6+1+2 & 1+2+2 \\ -6-6-6 & 1+2+1 & 2+2+2 \\ -6-6-6 & 2+1-6 & 2+2+1 \end{pmatrix} = \begin{pmatrix} -18 & -3 & 5 \\ -18 & 4 & 6 \\ -18 & -3 & 5 \end{pmatrix}$$

$$\boldsymbol{W}_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{X}_2 = \begin{pmatrix} 9 & 9 & 1 & 1 & 2 \\ 9 & 1 & 1 & 2 & 3 \\ 9 & 1 & 1 & 3 & 3 \\ 9 & 1 & 1 & 2 & 3 \\ 9 & 9 & 1 & 1 & 2 \end{pmatrix} \quad \boldsymbol{W}_{22} * \boldsymbol{X}_2 = \begin{pmatrix} 0+0+0 & 0+0+0 & 0+0+0 \\ 0+0+0 & 0+0+0 & 0+0+0 \\ 0+0+0 & 0+0+0 & 0+0+0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{W}_{23} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{X}_3 = \begin{pmatrix} 9 & 9 & & & \\ 9 & & & & 1 \\ 9 & & & & 1 \\ 9 & & & & 1 \\ 9 & 9 & & & \end{pmatrix} \quad \boldsymbol{W}_{23} * \boldsymbol{X}_3 = \begin{pmatrix} 0+0+0 & 0+0+0 & 0+0+0 \\ 0+0+0 & 0+0+0 & 0+0+0 \\ 0+0+0 & 0+0+0 & 0+0+0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

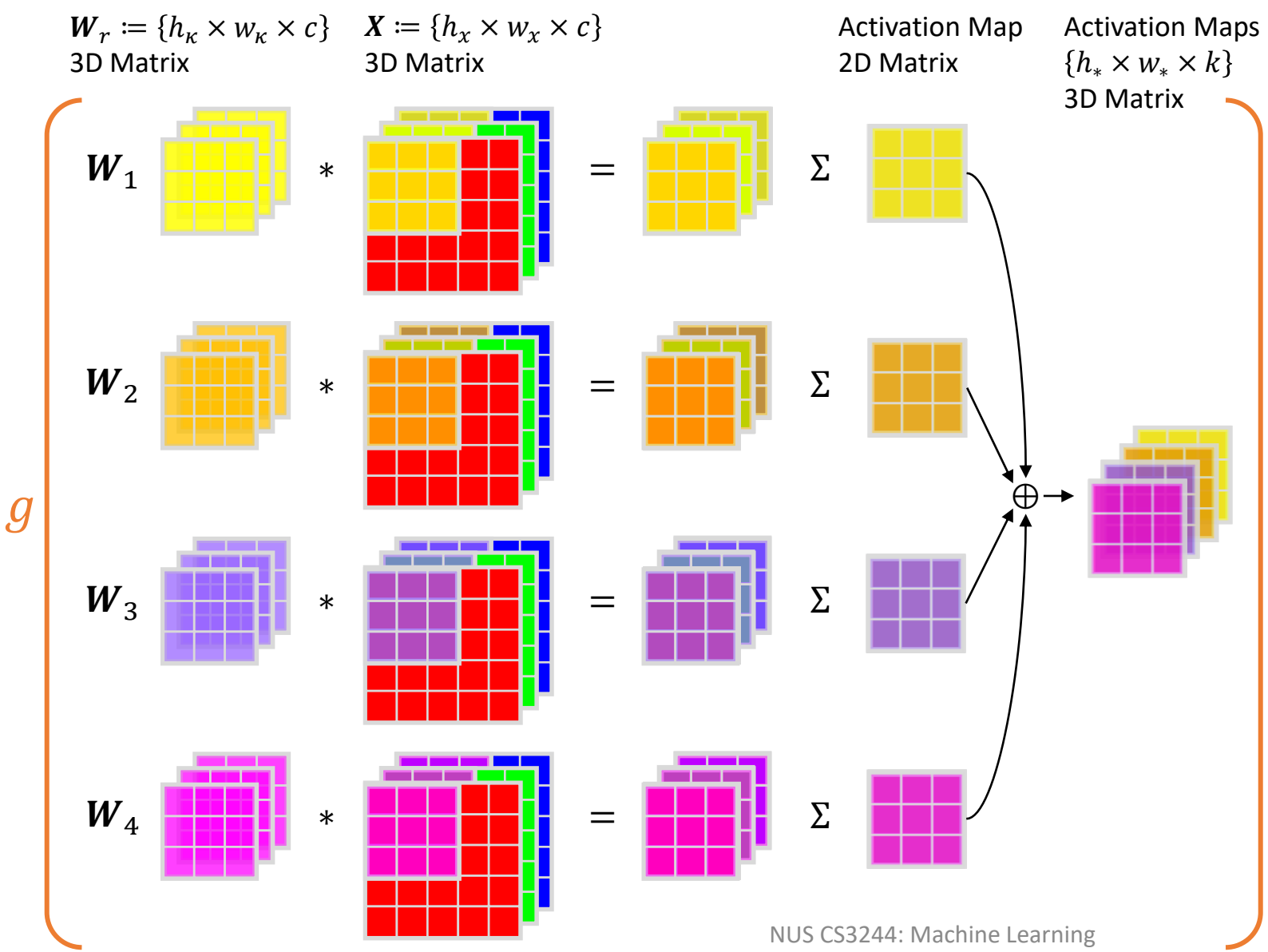
$$\rightarrow + \rightarrow \sum_{r=1}^{c=3} \boldsymbol{W}_{2r} * \boldsymbol{X}_r = \begin{pmatrix} -18 & -3 & 5 \\ -18 & 4 & 6 \\ -18 & -3 & 5 \end{pmatrix}$$

Kernels can be **different** for **different channels**.
In this case, vertical edge detector for red channel only


?
Questions!

have a

Multiple Convolution Kernels ($c = 3$ channels, $k = 4$ filters)



Don't confuse
- **Channels** of input image $\boldsymbol{x}$
- **Kernels/filters** of weights $\boldsymbol{W}$

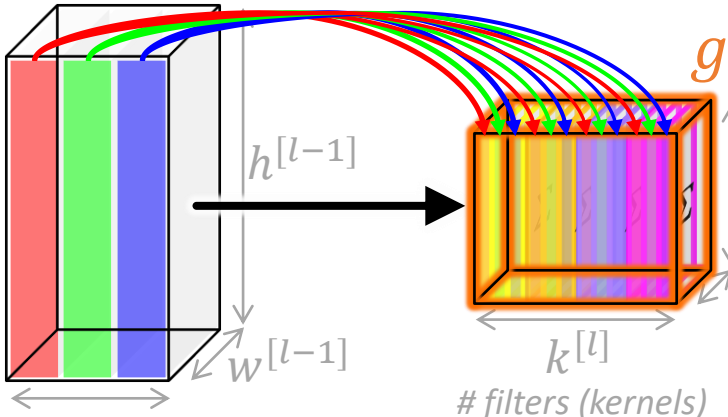
Key Takeaways
- Each Kernel convolves on **all channels** of image
- Each Activation Map is summed across channels
- 1 Activation Map per Kernel
- Kernel indexed in 3rd dimension of Activation Maps

Multi-Channel Multi-Kernel Convolutions

$$\boldsymbol{W}^{[l]} = \begin{pmatrix} \boldsymbol{W}_{11} & \boldsymbol{W}_{21} & \boldsymbol{W}_{31} & \boldsymbol{W}_{41} \\ \boldsymbol{W}_{12} & \boldsymbol{W}_{22} & \boldsymbol{W}_{32} & \boldsymbol{W}_{42} \\ \boldsymbol{W}_{13} & \boldsymbol{W}_{23} & \boldsymbol{W}_{33} & \boldsymbol{W}_{43} \end{pmatrix}$$

$\boldsymbol{W}^{[l]}$ 4D Matrix

$\left\{h_{\kappa}^{[l]} \times w_{\kappa}^{[l]} \times c^{[l-1]} \times k^{[l]}\right\}$



$$\boldsymbol{A}^{[l]} = g^{[l]}\left(\sum_{r}^{c^{[l-1]}} \boldsymbol{W}_r^{[l]} * \boldsymbol{A}_r^{[l-1]}\right)$$

$\boldsymbol{A}^{[l-1]}$ 3D Matrix

$\left\{h^{[l-1]} \times w^{[l-1]} \times c^{[l-1]}\right\}$

$\boldsymbol{A}^{[l]}$ 3D Matrix

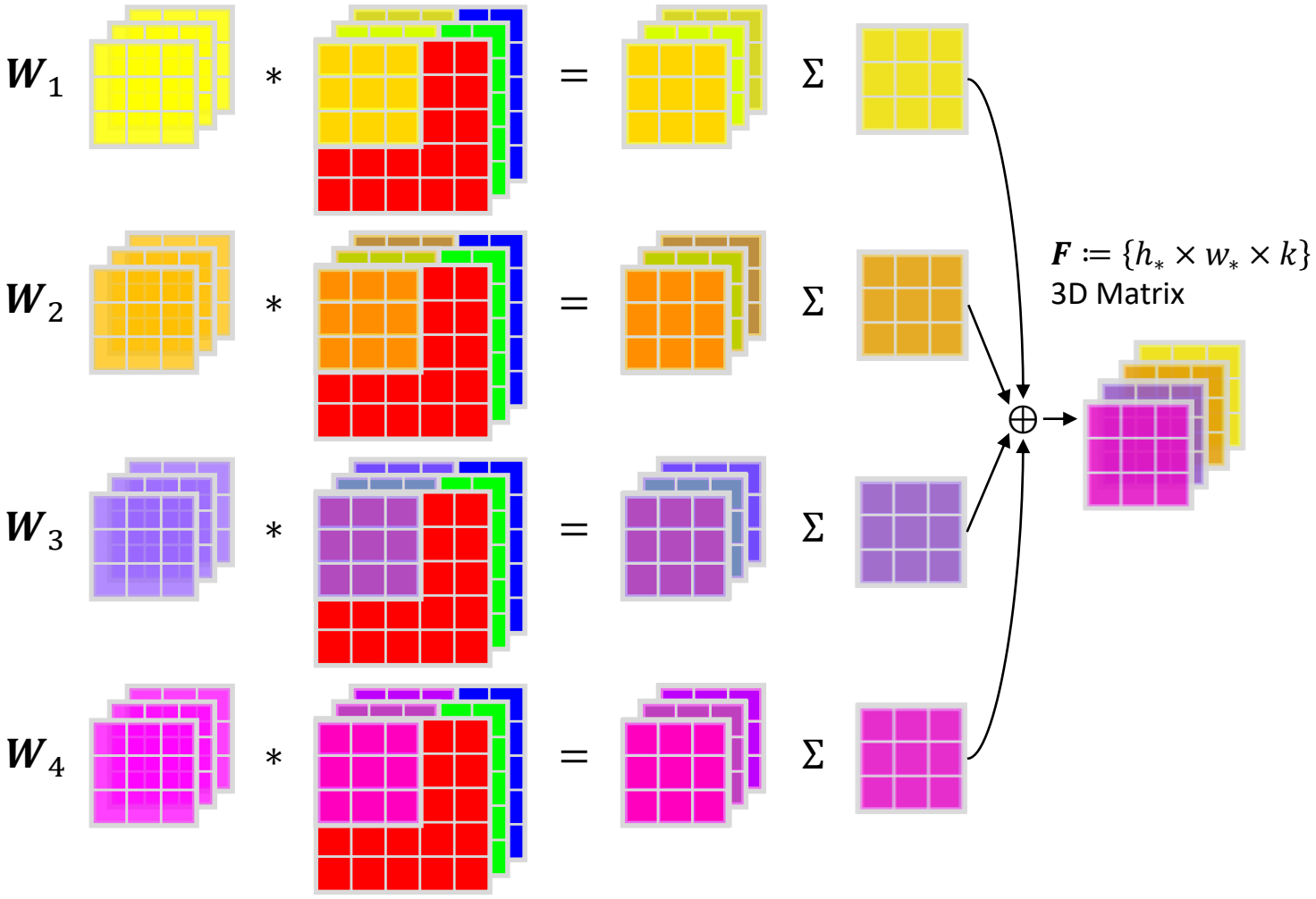
$\left\{h^{[l]} \times w^{[l]} \times k^{[l]}\right\}$

$h^{[l]}$ and $w^{[l]}$ calculated based on conv hyperparameters [E10a.1]

Multi-Channel Convolutions

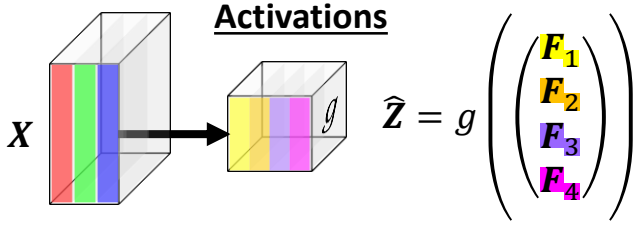
$\boldsymbol{W}_r \coloneqq \{h_\kappa \times w_\kappa \times c\}$
3D Matrix

$\boldsymbol{X} = \{h_x \times w_x \times c\}$
3D Matrix



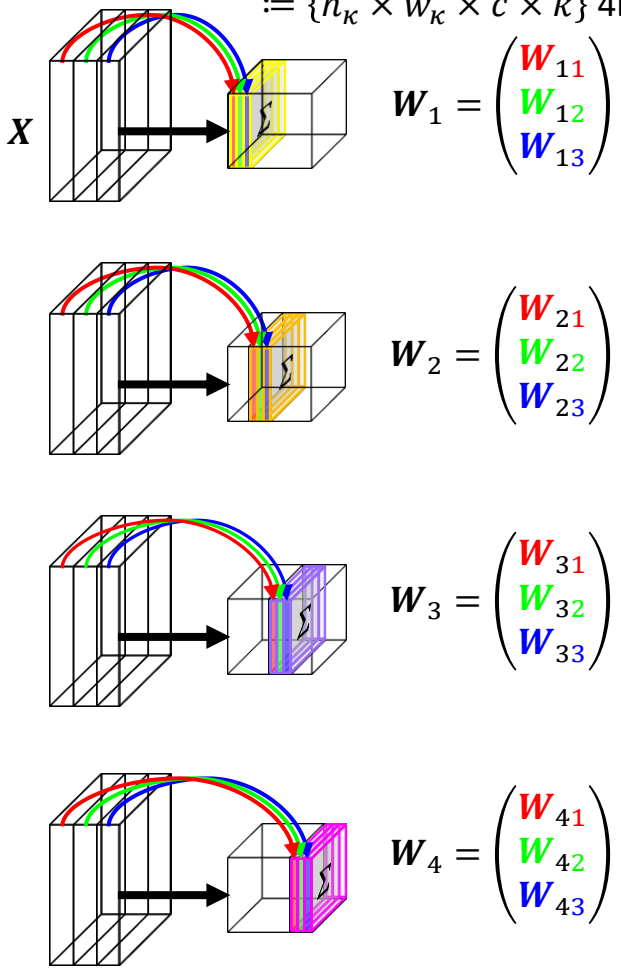
$\boldsymbol{F} \coloneqq \{h_* \times w_* \times k\}$
3D Matrix

Further reading: Andrew Ng lecture. Convolutions over Volumes



Weights

$\boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4)$
$\coloneqq \{h_\kappa \times w_\kappa \times c \times k\}$ 4D Matrix



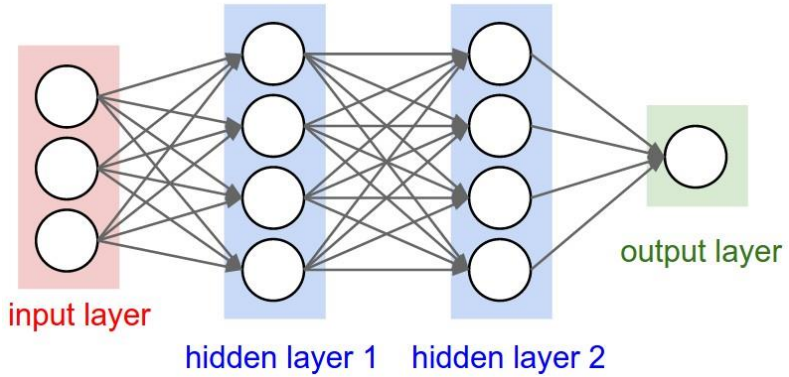
FC vs. Conv Layers

Remember: each kernel is like a different lens filter

Fully Connected (FC) Layers

Each layer has multiple activations ◯

- Each activation is a **0D scalar**
- Layer (of multiple activations) is a **1D vector**



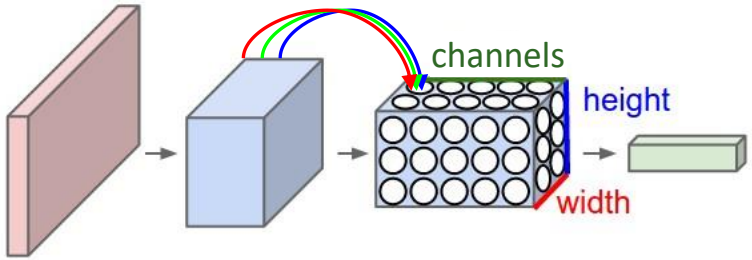
Weights →

- All weights connect activations of previous layer (1D) to current layer (1D)
- All weights represented as a **2D vector**

Convolutional (Conv) Layers

Each layer has multiple activation maps ▢

- Each activation map is a **2D matrix**
- Each map is on a different *channel* (1D)
- Layer is a **3D matrix**

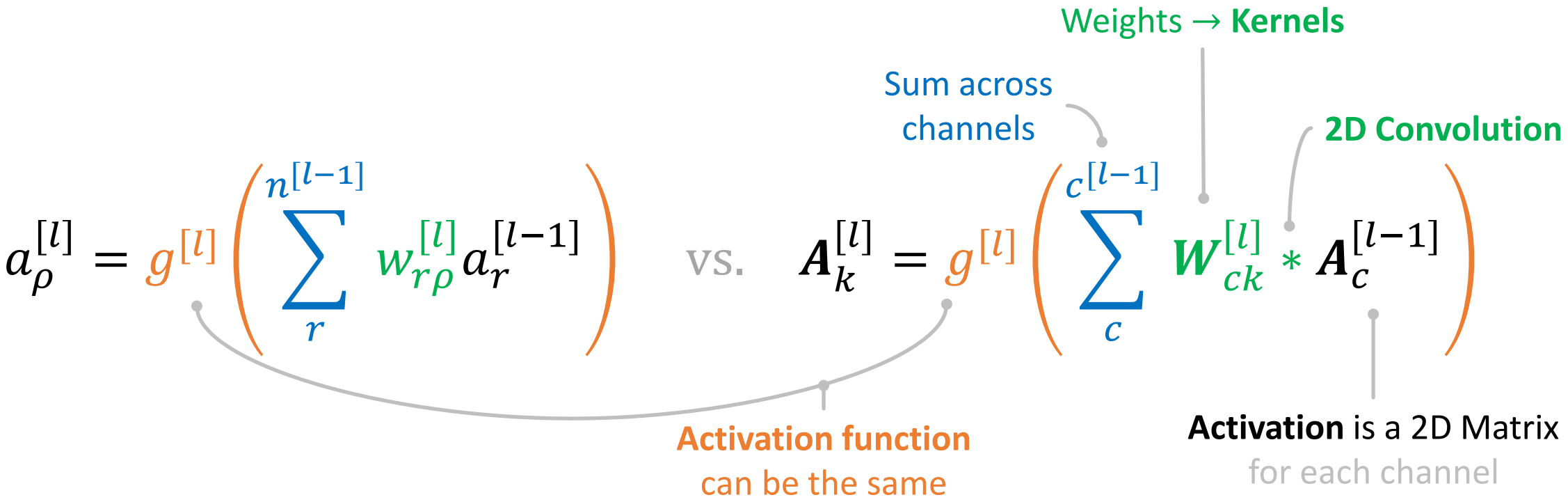


Kernels ▢ ↷

- Convolves on activation map (2D) of *all channels* (1D) in previous layer, then summed
- Each kernel represented as a **3D vector**
- Each kernel stored as separate *filters* (1D)
- All kernels represented as **4D vector**

Image credit: https://cs231n.github.io/convolutional-networks/

Fully-Connected Neuron vs. Convolutional Activation Map



Layer stores activation of **each neuron** ρ separately

Layer stores activation map of **each kernel** k separately

Convolutional Layer:
Feature Kernels & Feature Maps

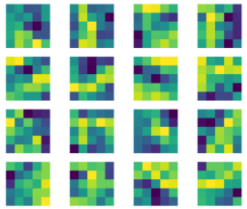
$$k^{[l-1]} = c^{[l]}$$
filters from previous layer $k^{[l-1]}$ is equal to #channels into current layer $c^{[l]}$

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

1 channel

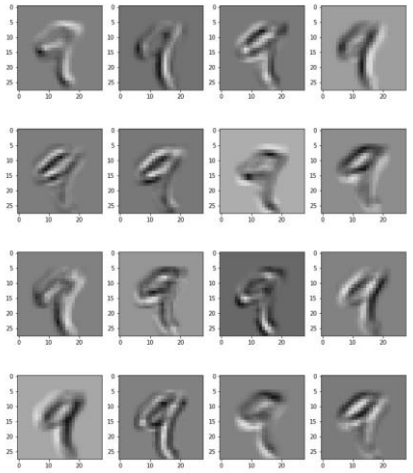
Input

16 filters



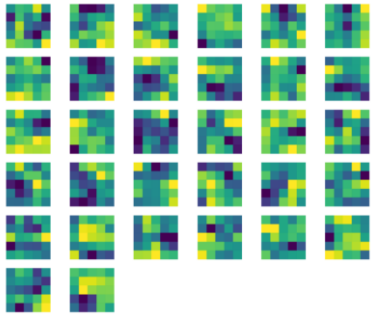
Layer 1 Feature Kernels

16 channels



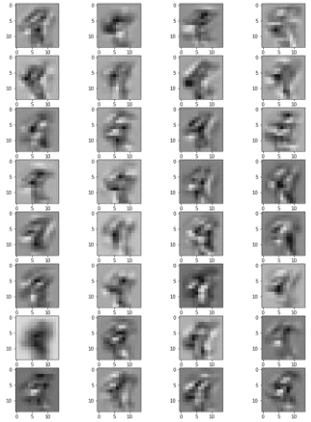
Layer 1
Activation Maps

32 filters



Layer 2
Feature Kernels

32 channels



Layer 2
Activation Maps

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**.

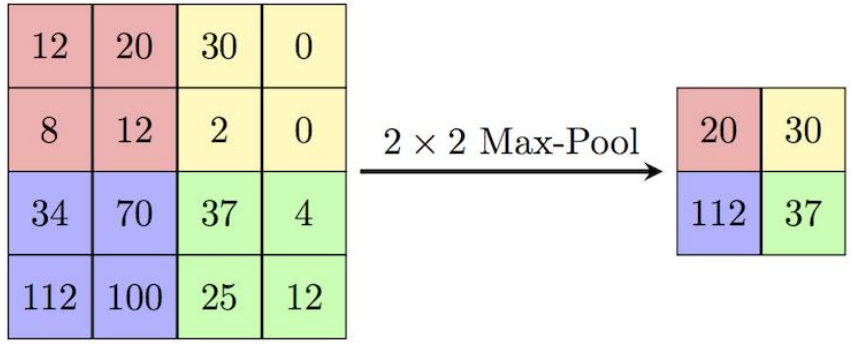
Interpretability: do you know what these kernel mean?

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

Pooling Layer

- **Downsamples** Feature Maps
- Helps to train later kernels to detect **higher-level** features
- Reduces **dimensionality**
- Aggregation methods
  - Max-Pool (most used)
  - Average-Pool
  - Sum-Pool

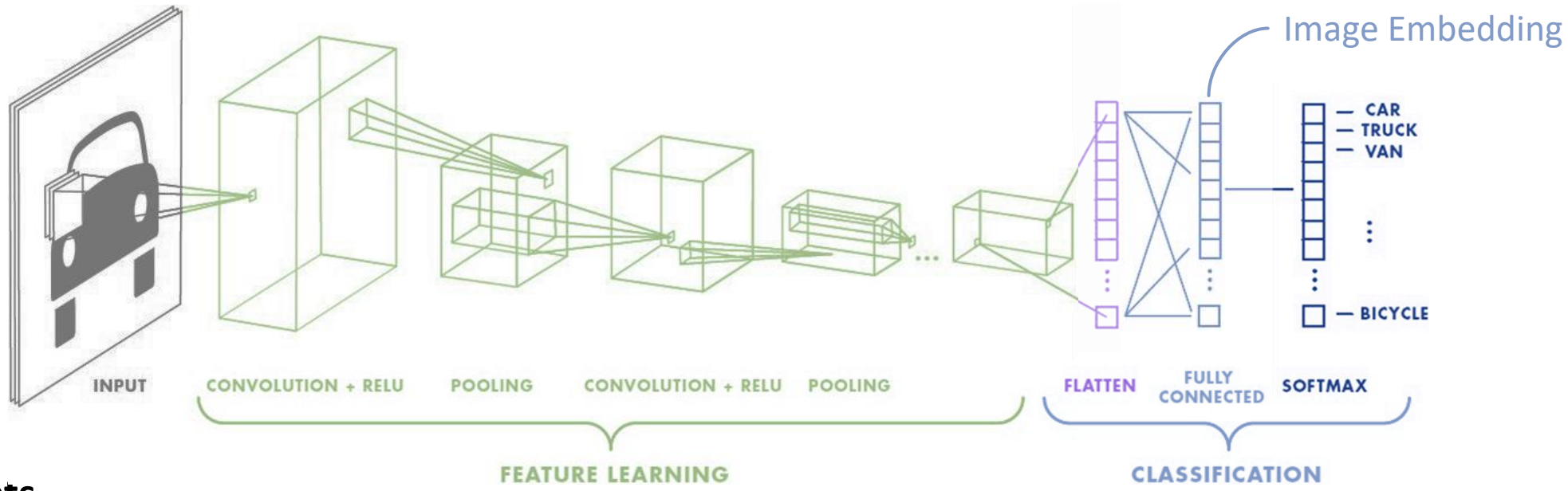
Calculation



Example

Image credit: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Convolutional Neural Network



Key concepts

❶ Learn Spatial Feature

- Series of multiple convolution + pooling layers
- Progressively learn more diverse and higher-level features
- Analogy: human visual cortex

❷ Flattening

- Convert to fixed-length 1D vector

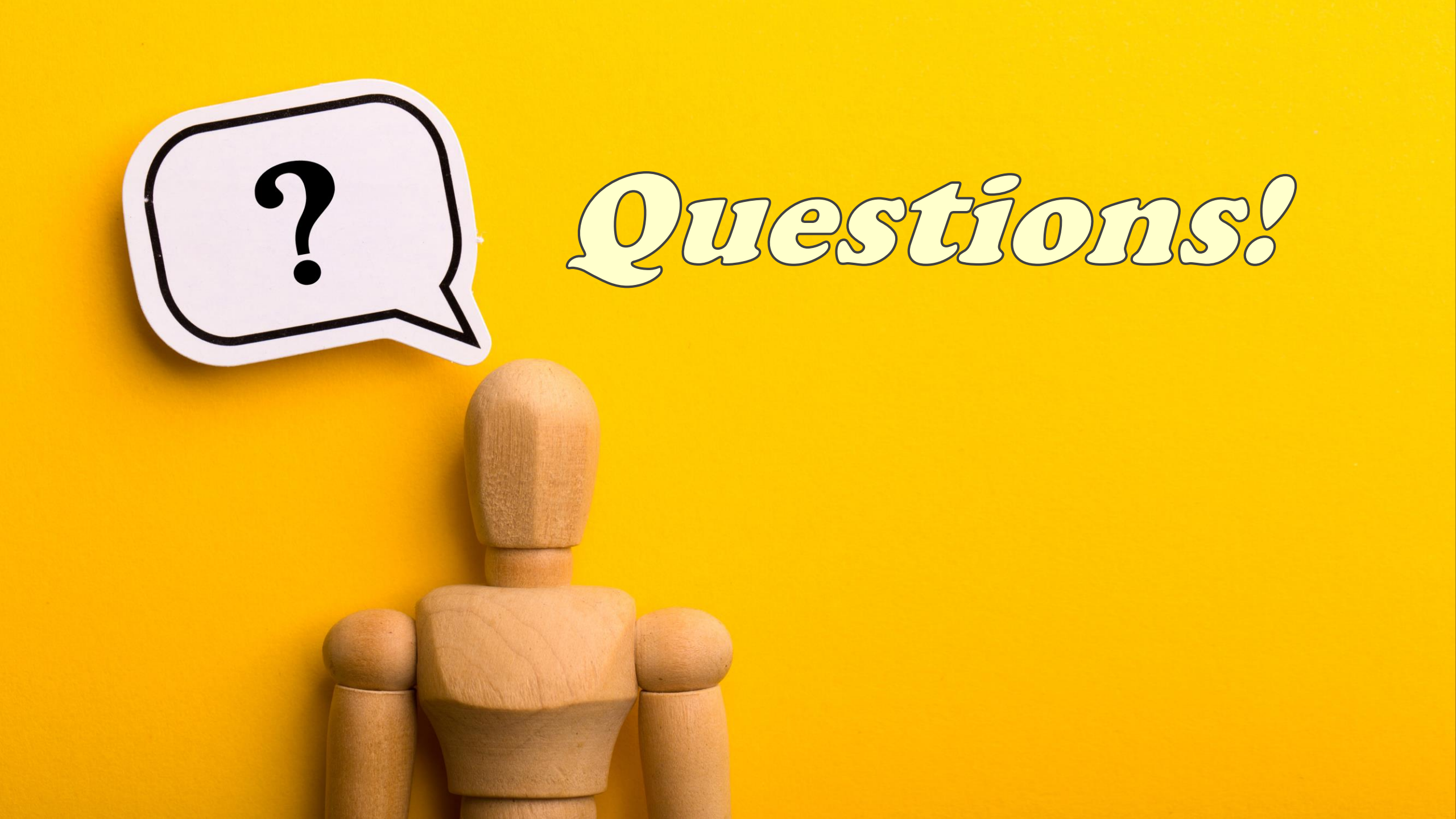
❸ Learn Nonlinear Features

- With fully connected layers (regular neurons)
- Learns nonlinear relations with multiple layers
- Analogy: semantic reasoning

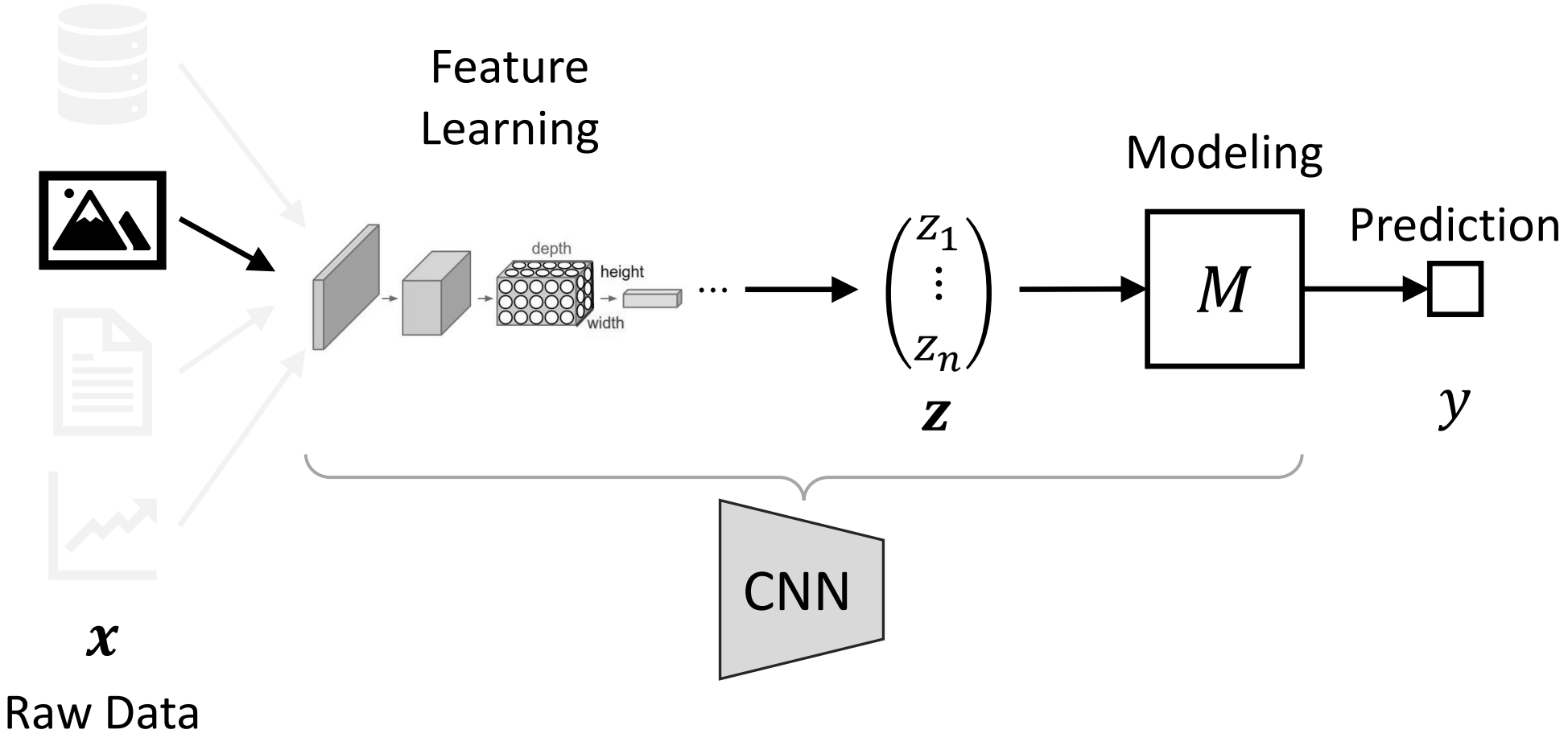
❹ Classification

- Softmax ≔ Multiclass Logistic Regression
- Feature input = image embedding vector (typically large vector)
- Analogy: decision making

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

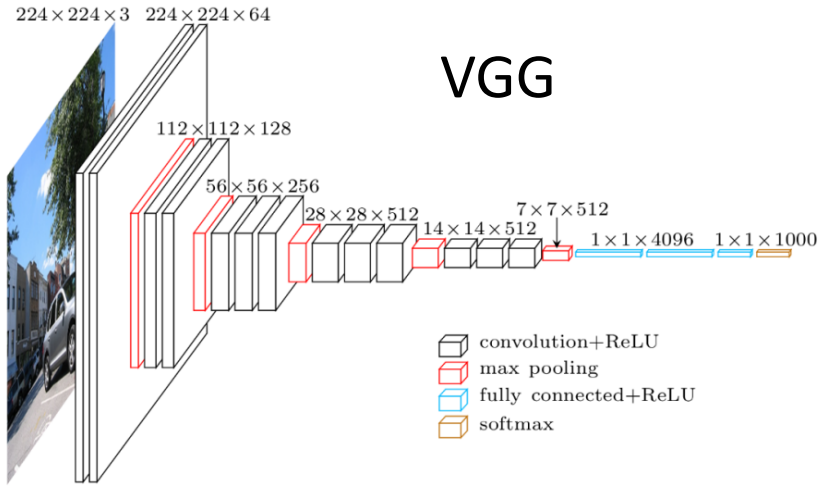

?
Questions!

From Manual Feature Engineering To Automatic Feature Learning

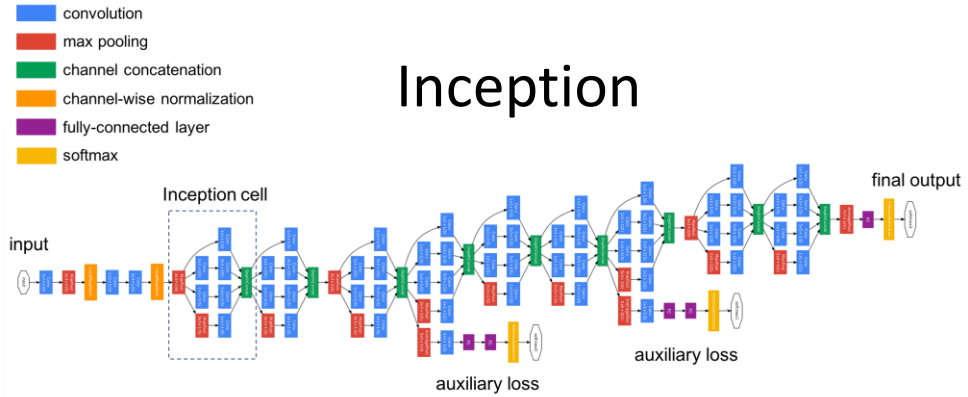


Other popular CNN architectures

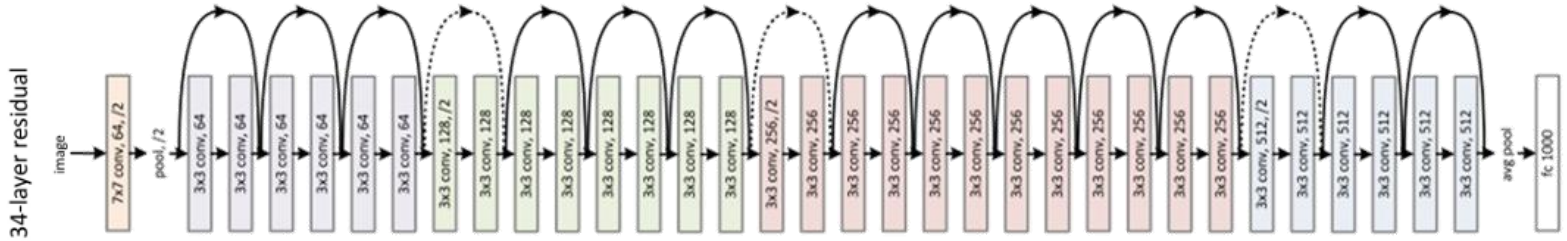
VGG



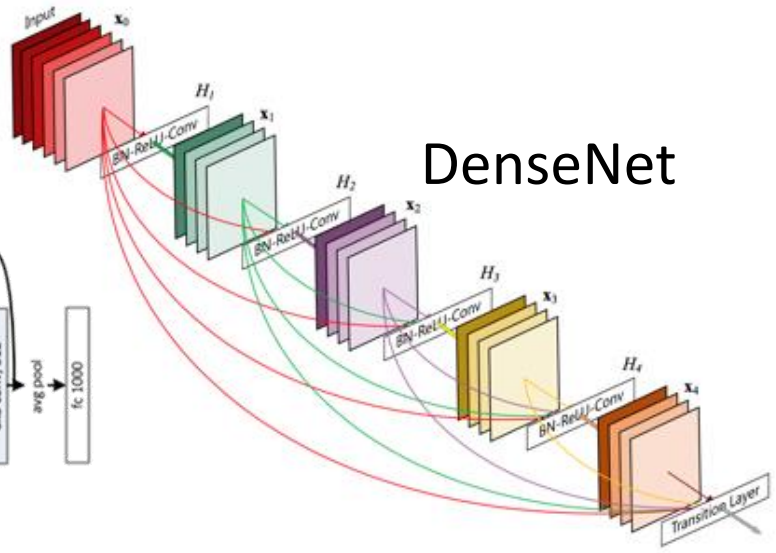
Inception



ResNet

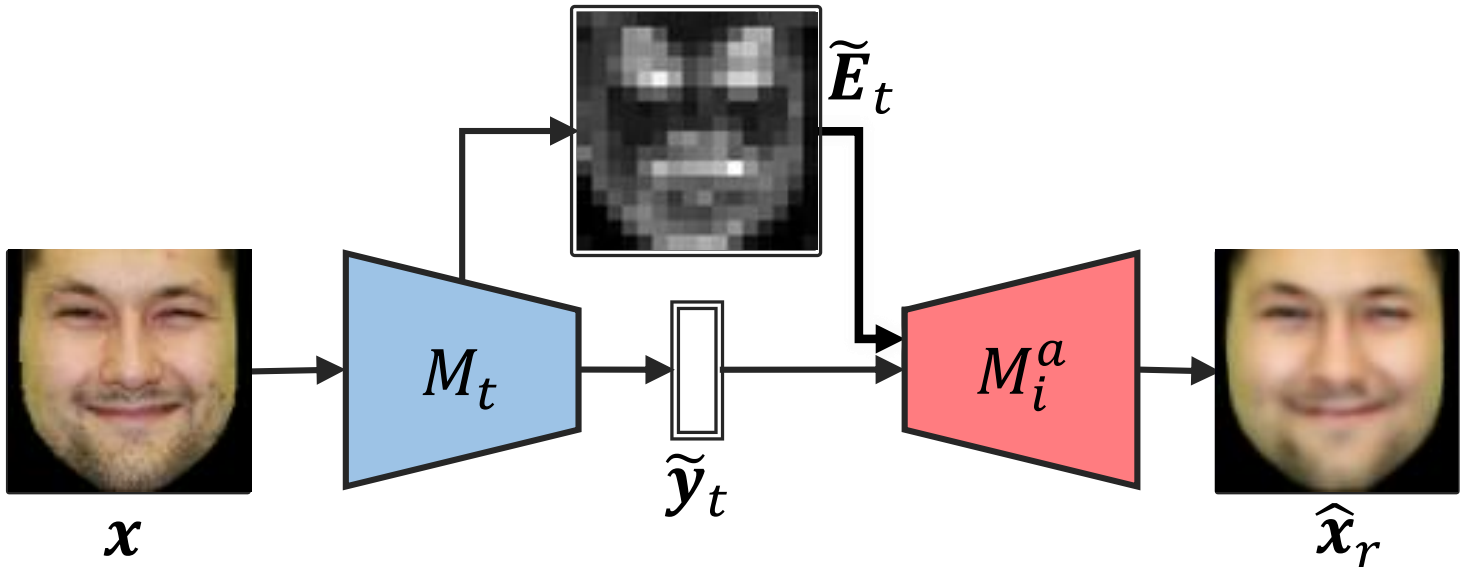


DenseNet



Further reading: https://www.jeremyjordan.me/convnet-architectures/

Model Inversion Attack:
Predicting Images from Classification Vector

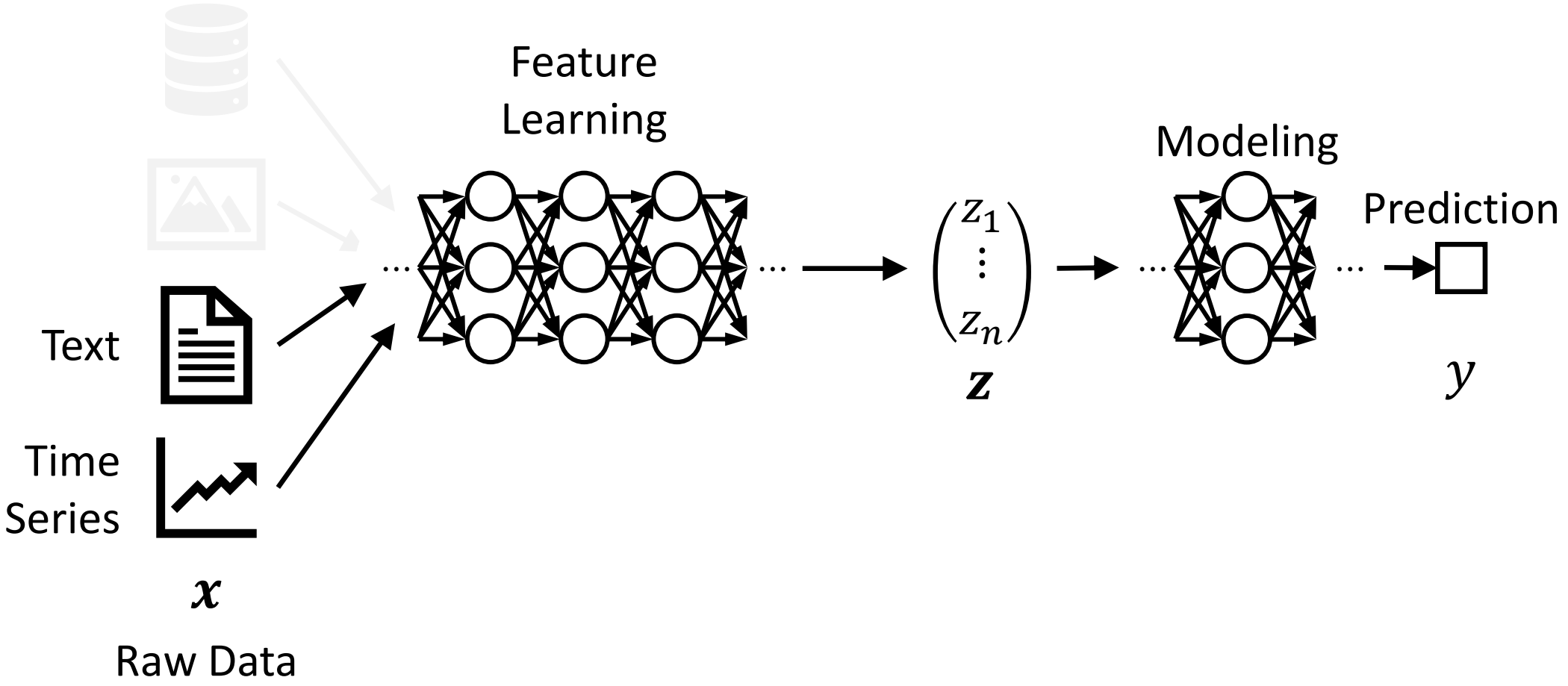


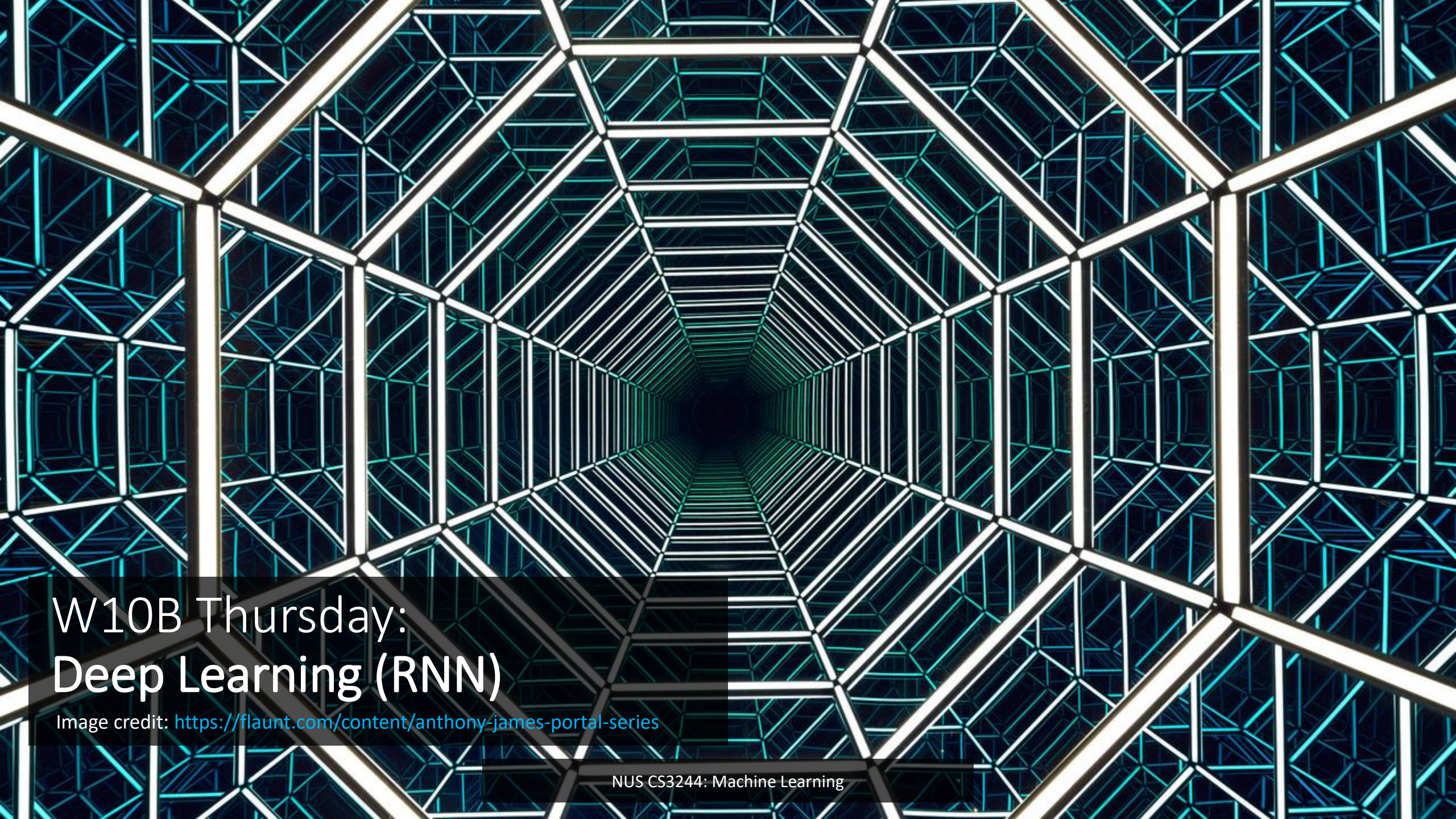
Model inversion attacks can reconstruct private face photos from prediction vectors only.

Model explanations can **worsen** model inversion attacks.

Zhao, X., Zhang, W., Xiao, X., & Lim, B. Y. (2021). Exploiting Explanations for Model Inversion Attacks. *ICCV 2021*.

From Manual Feature Engineering To Automatic Feature Learning





W10B Thursday:
Deep Learning (RNN)

Image credit: https://flaunt.com/content/anthony-james-portal-series