Pattern Recognition
Lecture 04-1

Basic Deep Learning

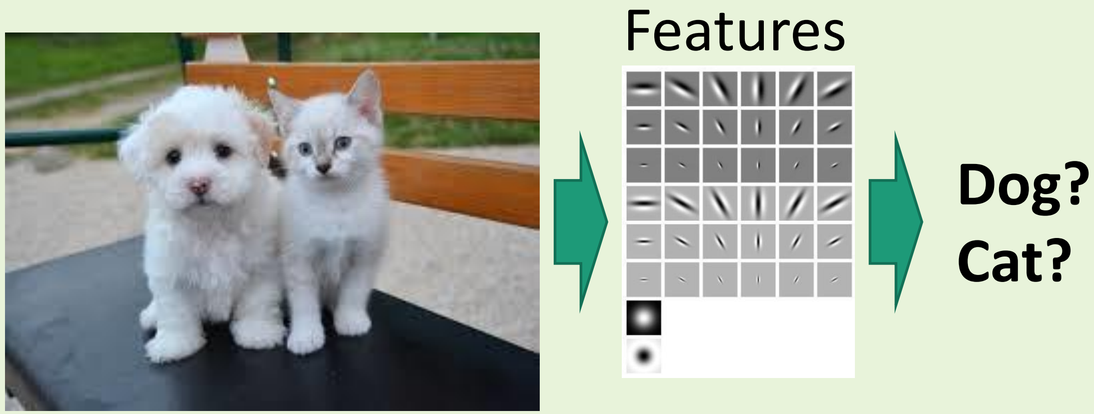
Prof. Jongwon Choi
Chung-Ang University
Fall 2022

This Class

- **Supervised Deep Learning**
  - Definition
  - Architecture
  - Prediction
  - Training
- **Unsupervised Deep Learning – Auto-encoder**
  - Definition
  - Architecture
  - Prediction
  - Training

Supervised Deep Learning - Definition

- **Limitation 1 of the traditional ML**
  - Very hard to classify the **<u>complex non-linearity</u>** of sample features
  - For example, classify the class labels from the image!
  - Thus, we've utilized the features like LoG filter and filter banks

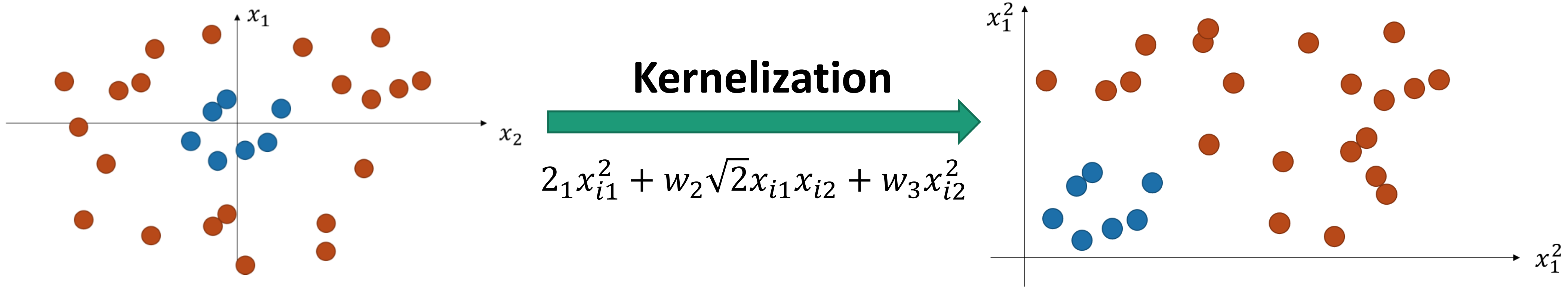


The traditional ML

Impossible by the traditional ML

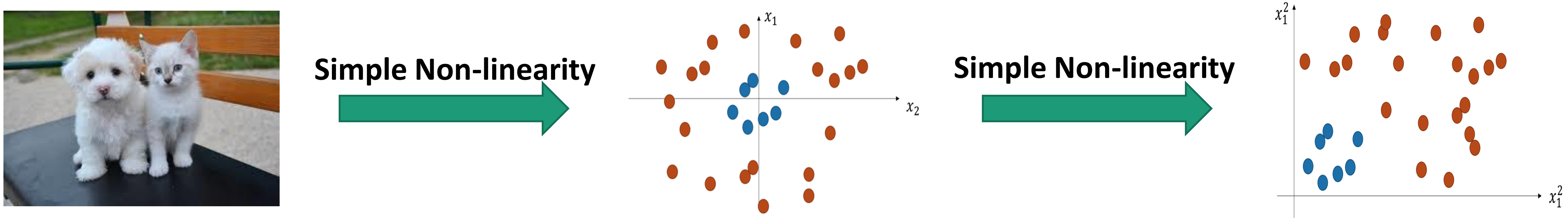
Supervised Deep Learning - Definition

- **Limitation 2 of the traditional ML**
  - To cover the complex non-linearity,
  - we can expand the complexity of kernelization,
  - which causes the increased number of parameters!
  - With the large number of parameters, **<u>overfitting problem</u>** happens



Supervised Deep Learning - Definition

- **Solution 1 of the deep learning – Complex non-linearity?**
  - Let’s iterate the simple non-linearity for the complex non-linearity
  - Thus, a neural network can represent the complex non-linearity.



Supervised Deep Learning - Definition

- **Solution 2 of the deep learning – Overfitting with large parameter?**
  - We can avoid the overfitting problem by large data!
  - Data Data Data! Many data is very very important!!

Data + **Large Neural Network**

Overfitting!!

Low training error

High test error

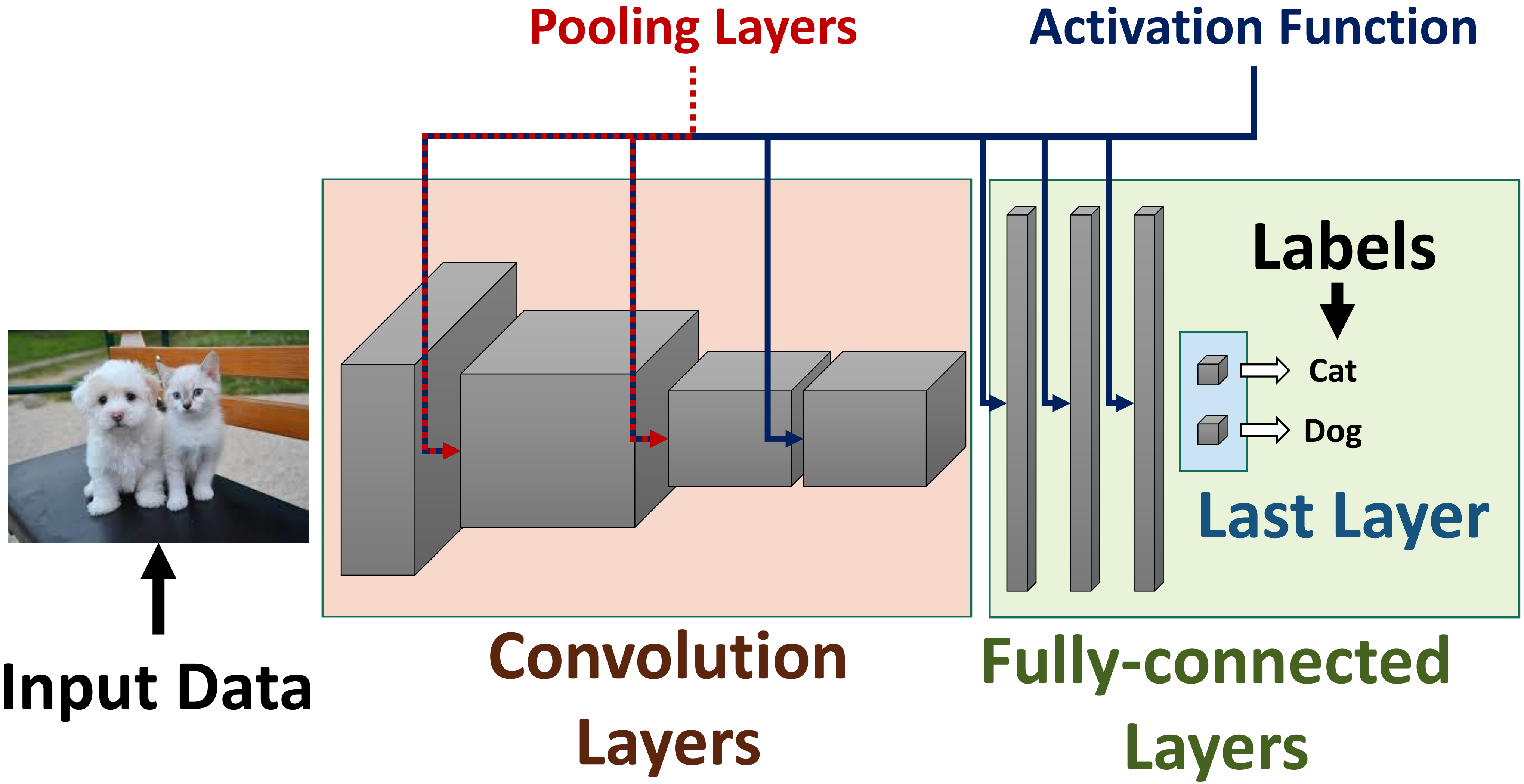
Data + **Large Neural Network**

What is Overfitting??

Low training error

Low test error

Supervised Deep Learning - Architecture



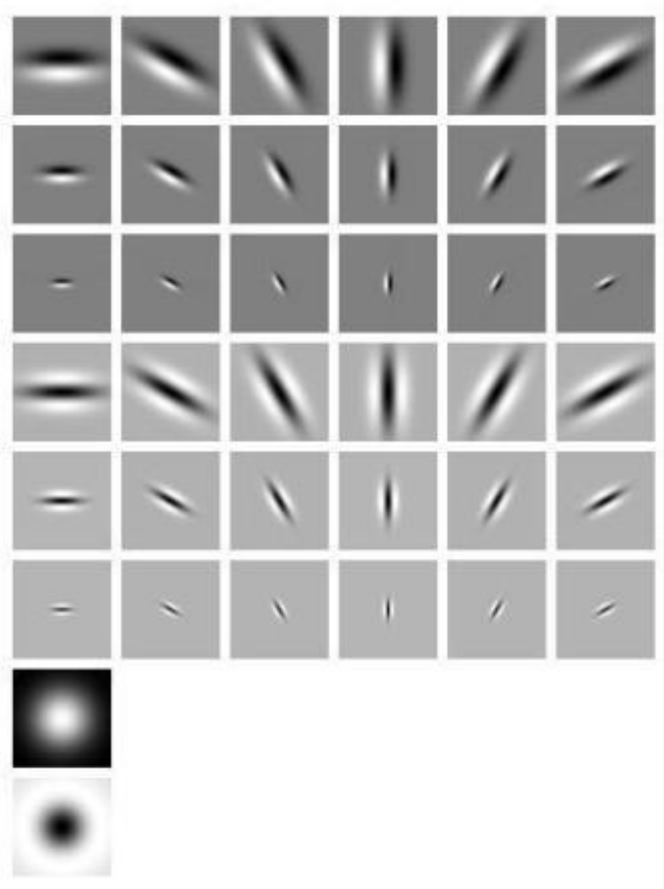
Feature Engineering – Convolution

- 1-D Convolution Examples – [0 1 1 2 3 5 8 13]
  - "Gaussian" – $\exp\left(-\frac{i^2}{2\sigma^2}\right)$
    - Blurring
  - "Sharpen convolution" - [-1 3 -1]
    - Average that places negative weights on the surrounding pixels
  - "Laplacian" – [-1 2 -1]
    - Sum to zero filters
    - Approximates second derivative!
  - "Laplacian of Gaussian Filter" - $\left(1-\frac{\mathrm{i}^2}{2\sigma^2}\right)\exp\left(-\frac{i^2}{2\sigma^2}\right)$

Images and Higher-Order Convolution

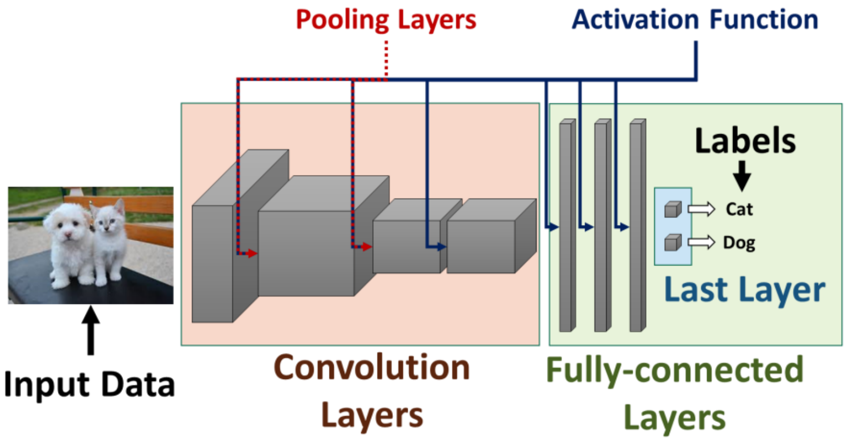
- Notable convolutions
  - Gaussian – blurring/Averaging
  - Laplace of Gaussian (LoG) – Second-derivative
  - Gabor filters – directional first- or higher-derivative

Filter Banks



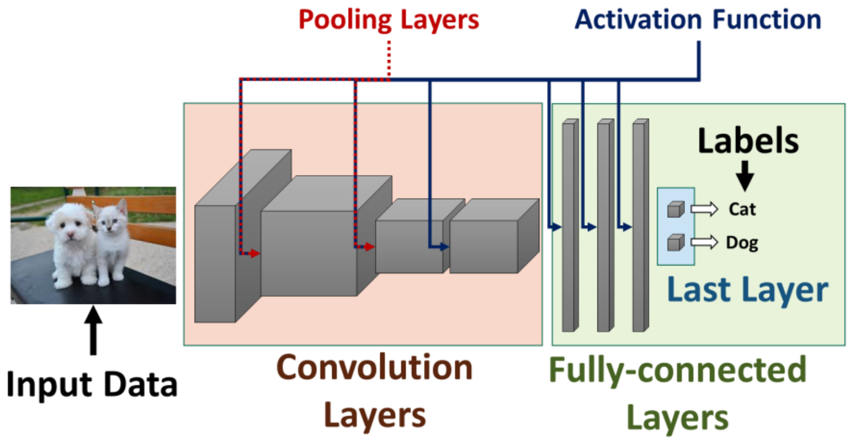
Supervised Deep Learning - Architecture

- **Convolution filter layer (Conv. Layer)**
  - Similar to the feature extraction filters.



Supervised Deep Learning - Architecture

- **Convolution filter layer (Conv. Layer)**
  - Similar to the feature extraction filters.
  - Difference



	Traditional Filters	Convolutional Layers
#(Filter Types)	10~30	64~1024
Filter Types	Predefined by users	Obtained by training
Target Values	Unnecessary	Unknown
Input of the Filters	Input image	Image/Filter response map

Hidden Feature
Latent Feature

Supervised Deep Learning - Architecture

- **Convolution filter layer (Conv. Layer)**
  - Similar to the feature extraction filters.

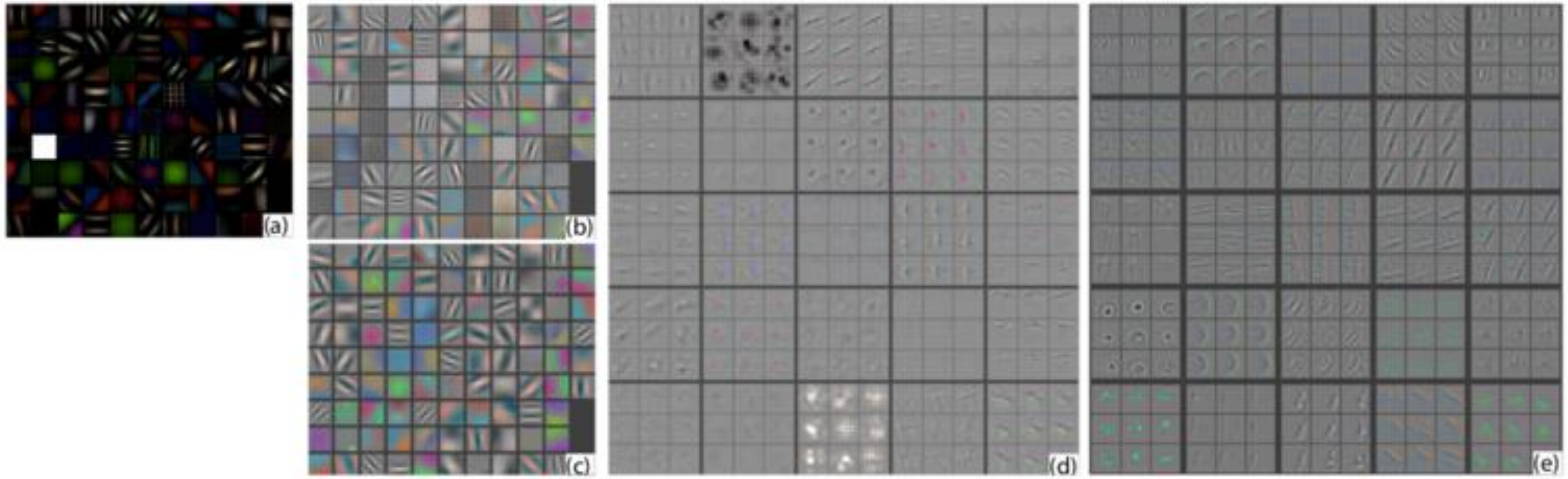
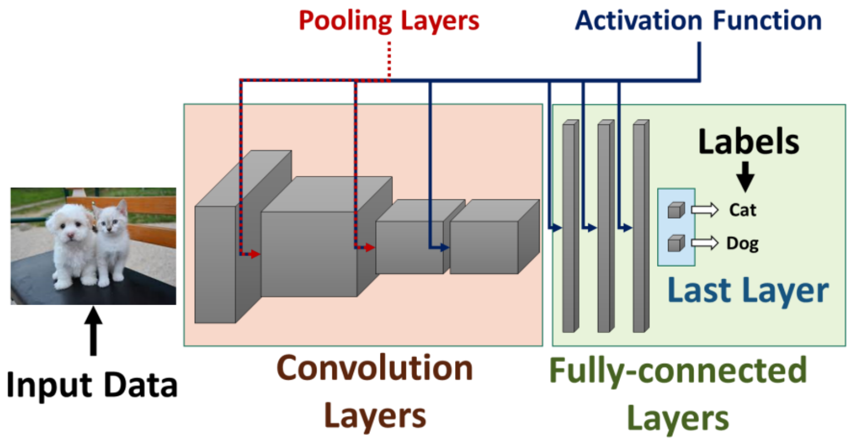


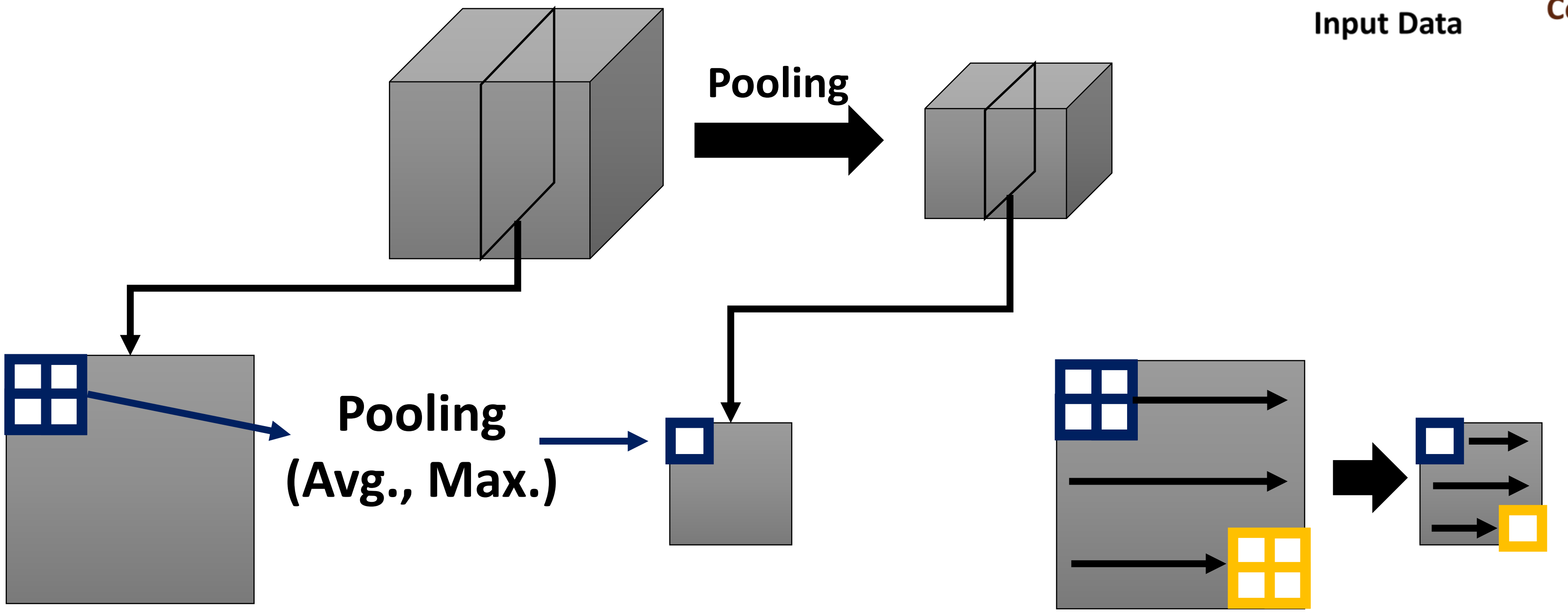
Figure 6. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from (Krizhevsky et al., 2012). (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from (Krizhevsky et al., 2012). (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

*** "ZFNet - Visualizing and understanding convolutional networks", ECCV2014

Supervised Deep Learning - Architecture

- **Pooling layer**
  - Reduce the resolution of feature maps





Linear Regression – 1 Dimension

- Assume that we only have 1 feature (d=1):
  - ex. x_i is the length of education and y_i is income
- Linear regression makes predictions $\hat{y}_i$ using a linear function of x_i:

$$\hat{y}_i = wx_i$$

  - The parameter ‘w’ is the weight or regression coefficient of x_i
  - As x_i changes, slope ‘w’ affects the rate that $\hat{y}_i$ increases/decreases:
    - Positive ‘w’: $\hat{y}_i$ increases as x_i increases
    - Negative ‘w’: $\hat{y}_i$ decreases as x_i increases

Multiple Dimension Linear Function

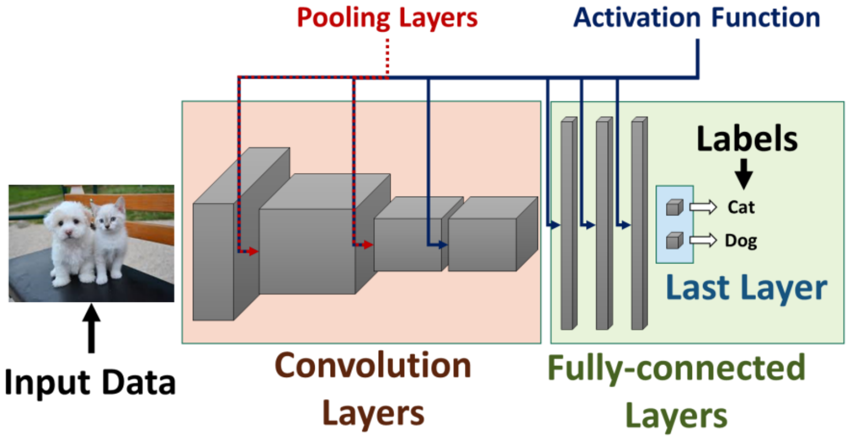
- A simple way is with a d-dimensional linear model
  - $\widehat{y_i} = w_1 x_{i1} + w_2 x_{i2} + w_3 x_{i3} + \cdots + x_d x_{id}$
  - In words, our model is that the output is a weighted sum of the inputs

- We can re-write this in summation notation:
  - $\widehat{y_i} = \sum_{j=1}^{d} w_j x_{ij}$

- We can also re-write this in vector notation: (inner product)
  - $\widehat{y_i} = \mathbf{w}^{\mathrm{T}} \mathbf{x}_i$
  - In this course, a vector is a column vector

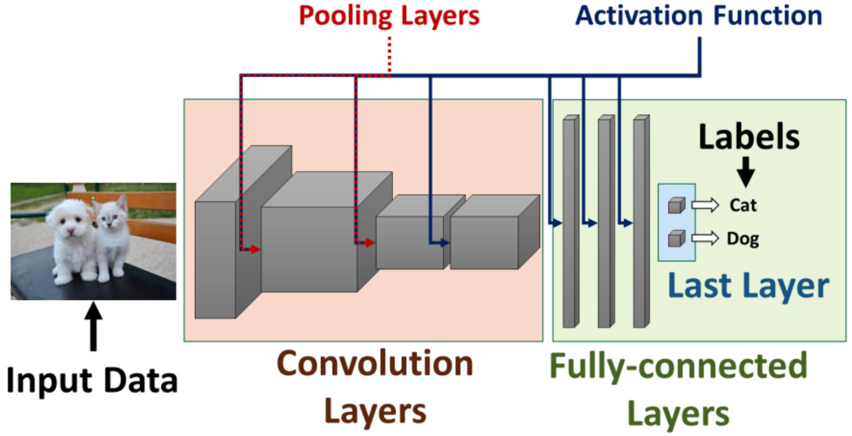
Supervised Deep Learning - Architecture

- **Fully-connected layer (FC Layer)**
  - Similar to multi-dim. linear regression models!



Supervised Deep Learning - Architecture

- **Fully-connected layer (FC Layer)**
  - Similar to multi-dim. linear regression models!
  - Difference



	Basic Linear Regressor	FC Layer
#(Output)	1	128~2048
Target	Valued labels	Unknown (Except the last one)
Input	Input features	Input features / Layer response map

Hidden Feature
Latent Feature

Linear Model Classification

- Binary classification using regression
  - Set $y_i = +1$ for one class
  - Set $y_i = -1$ for the other class

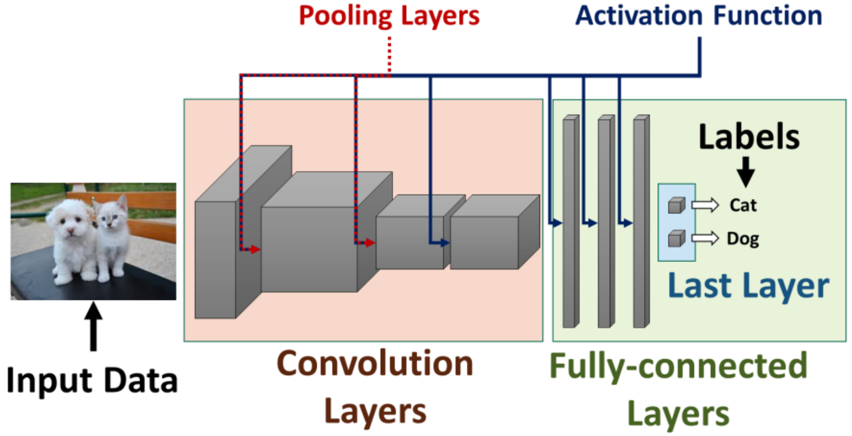
- At training time, fit a linear regression model:
  - $\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} = \mathbf{w}^T \mathbf{x}_i$
  - Then, the model will try to make $\mathbf{w}^T \mathbf{x}_i = +1$ for one class, and $\mathbf{w}^T \mathbf{x}_i = -1$ for the other class

“One vs All” Linear Classification

- “One vs all” logistic regression for classifying as cat/dog/person
  - Train a separate classifier for each class
    - Classifier 1 tries to predict +1 for “cat” images and -1 for “dog” and “person” images
    - Classifier 2 tries to predict +1 for “dog” images and -1 for “cat” and “person” images
    - …
  - Results in a weight vector w_c for each class ‘c’:
    - Weights w_c try to predict +1 for class ‘c’ and -1 for all others
    - We’ll use ‘W’ as a matrix with the w_c as rows

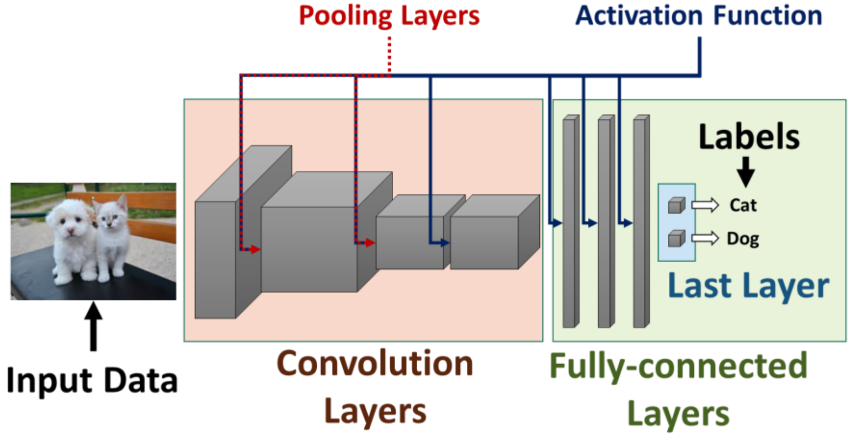
Supervised Deep Learning - Architecture

- **Last classification layer**
  - Similar to multi-class classification models!
  - One of FC layers



Supervised Deep Learning - Architecture

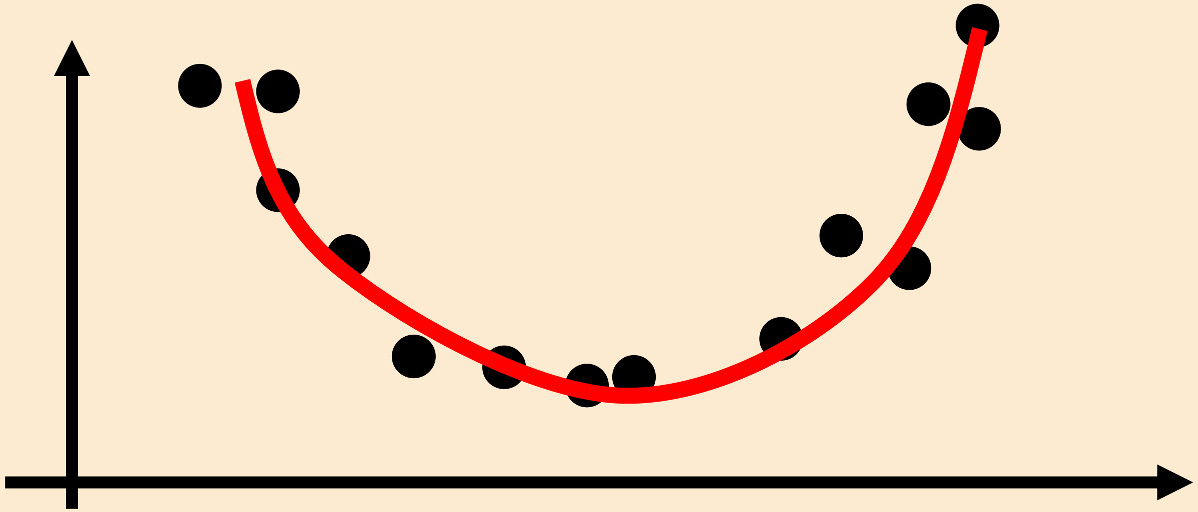
- **Last classification layer**
  - Similar to multi-class classification models!
  - One of FC layers
  - Difference



	Traditional Classification Model	Last Layer
Input	Handcraft features (Filter banks, etc.)	Layer response map

Non-linear Feature Transforms

- Can we use linear least squares to fit a quadratic model?
  - Yes! By adding a non-linear feature as:
  - $\hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2$
  - It's a linear function of w, but a quadratic function of x_i
  - $\hat{y}_i = \mathbf{v}^{\mathrm{T}} \mathbf{z}_i = v_1 z_{i1} + v_2 z_{i2} + v_3 z_{i3}$



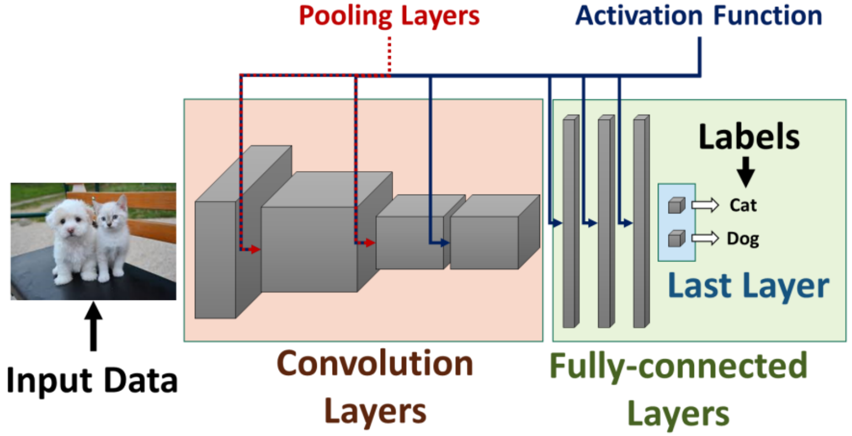
Kernel Trick

- Kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_i^T \mathbf{z}_j$
  - Instead of the acquisition of $\mathbf{z}_i$ and $\mathbf{z}_j$,
  - Directly estimate the Gram matrices from $\mathbf{x}_i\ and\ \mathbf{x}_j$

- Linear Kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j = \mathbf{z}_i^T \mathbf{z}_j$
- Polynomial Kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p = \mathbf{z}_i^T \mathbf{z}_j$
- Gaussian-RBF Kernel: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) = \mathbf{z}_i^T \mathbf{z}_j$

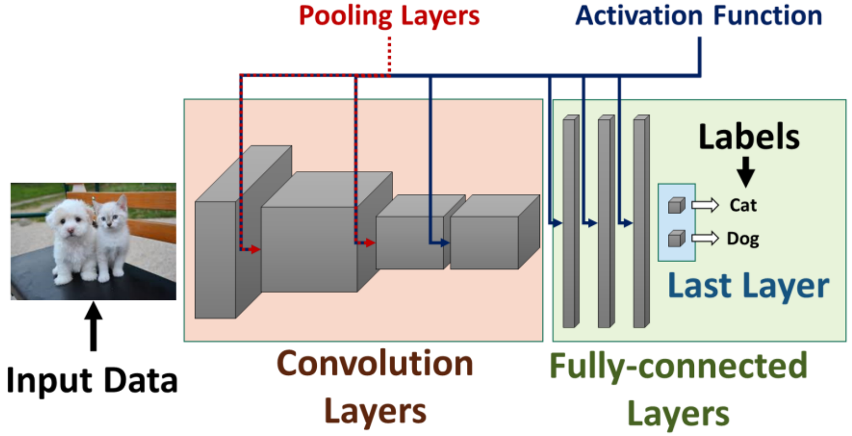
Supervised Deep Learning - Architecture

- **Activation function**
  - Similar to the kernelization!
  - Conventionally, always follows the layers



Supervised Deep Learning - Architecture

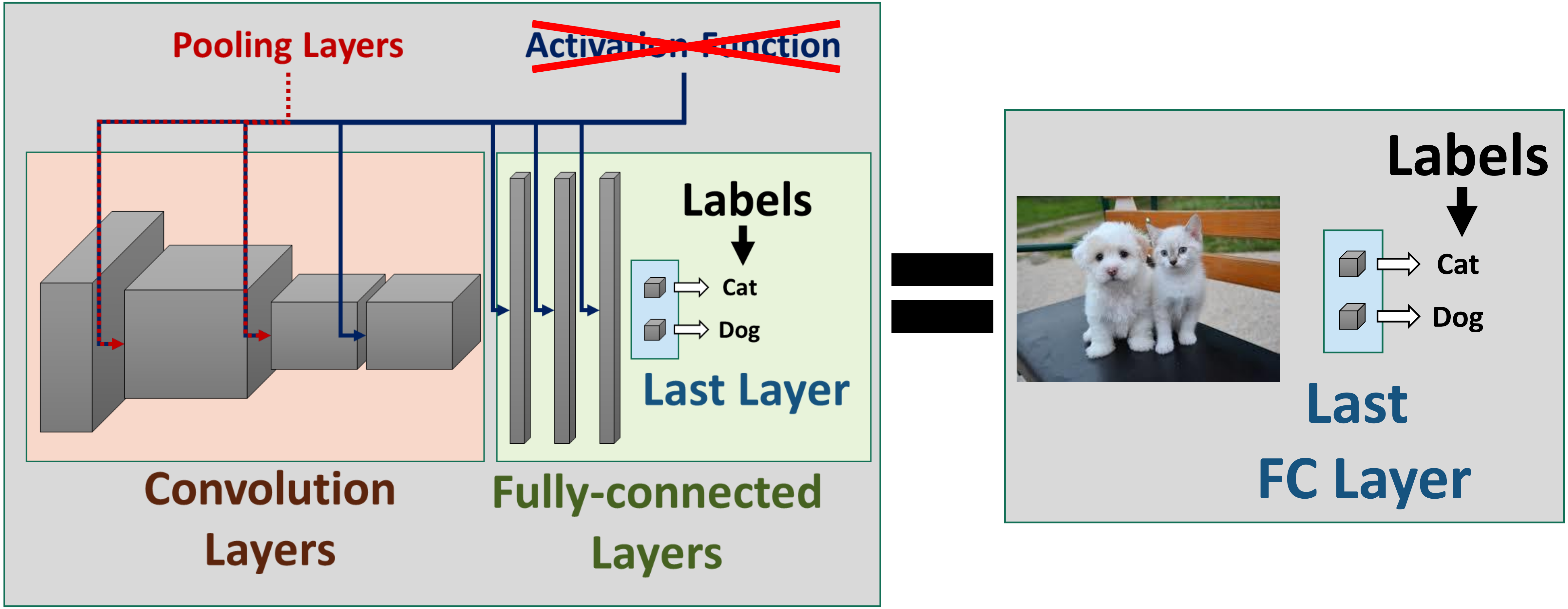
- **Activation function**
  - Similar to the kernelization!
  - Conventionally, always follows the layers
  - Difference



	Kernelization	**Activation function**
Motivation	Non-linear feature distribution	Neuro-science
Complexity	Depends on data	Fixed
#(Application)	1	1~1000

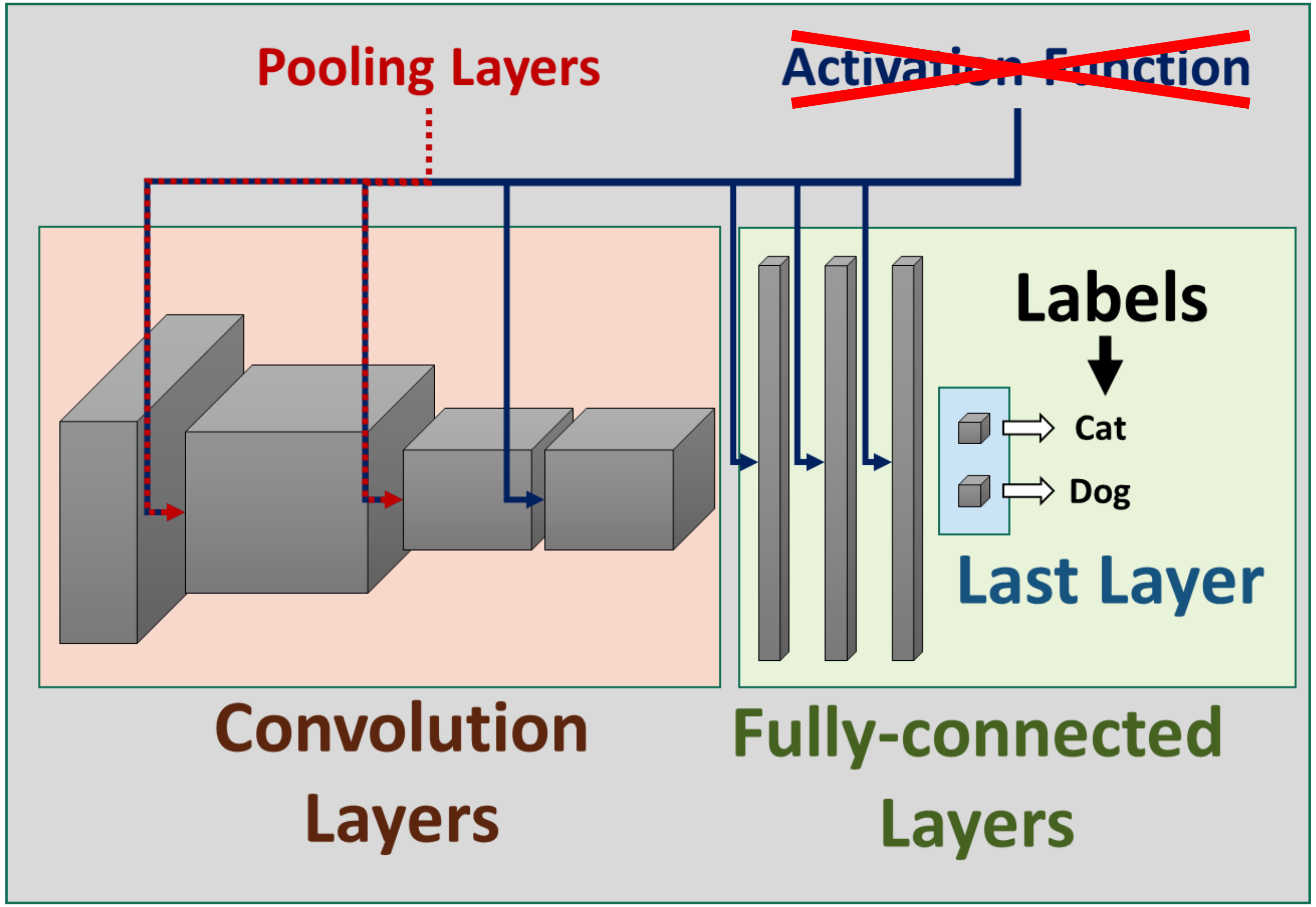
Supervised Deep Learning - Architecture

- **What happens without the activation functions?**

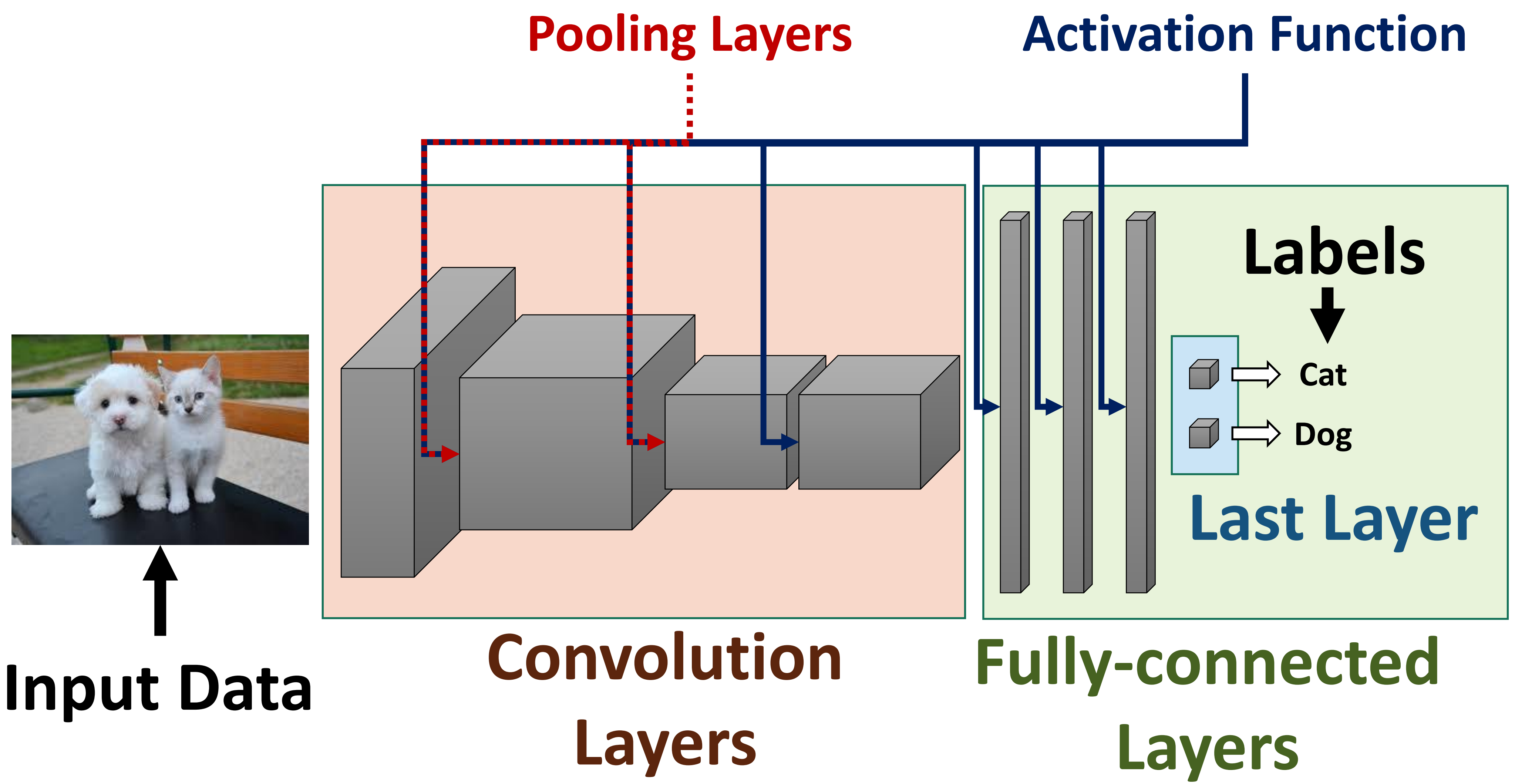


Supervised Deep Learning - Architecture

- **What happens without the activation functions?**

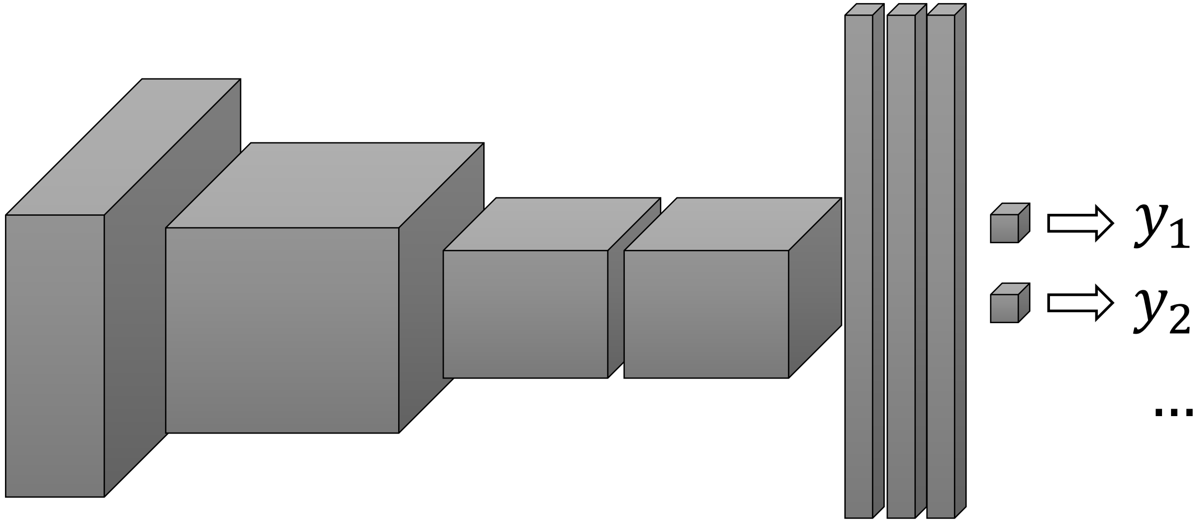


Supervised Deep Learning - Prediction



Supervised Deep Learning - Prediction

Input Data



Softmax

$$p(\hat{y}_1|\mathbf{x},\mathbf{y}) = \frac{\exp(y_1)}{\sum \exp(y_c)}$$

$$p(\hat{y}_2|\mathbf{x},\mathbf{y}) = \frac{\exp(y_2)}{\sum \exp(y_c)}$$

...

All Sum = 1