ARTIFICIAL INTELLIGENCE NEEDS REAL INTELLIGENCE

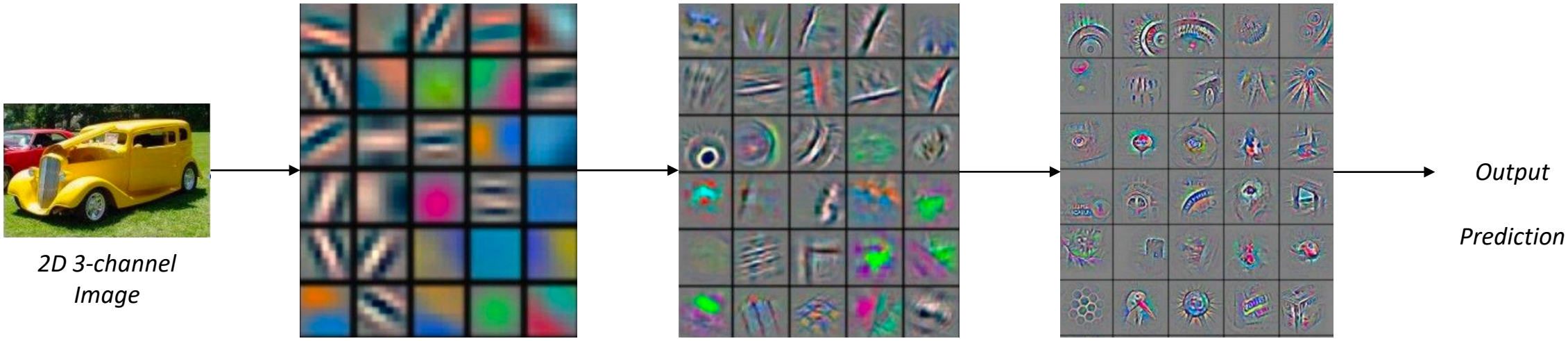
Convolutional Neural Network II

Professor: Tianyuan ZHANG
tianyuan.zhang@kedgebs.com

KEDGE BUSINESS SCHOOL

Recap of Previous Session

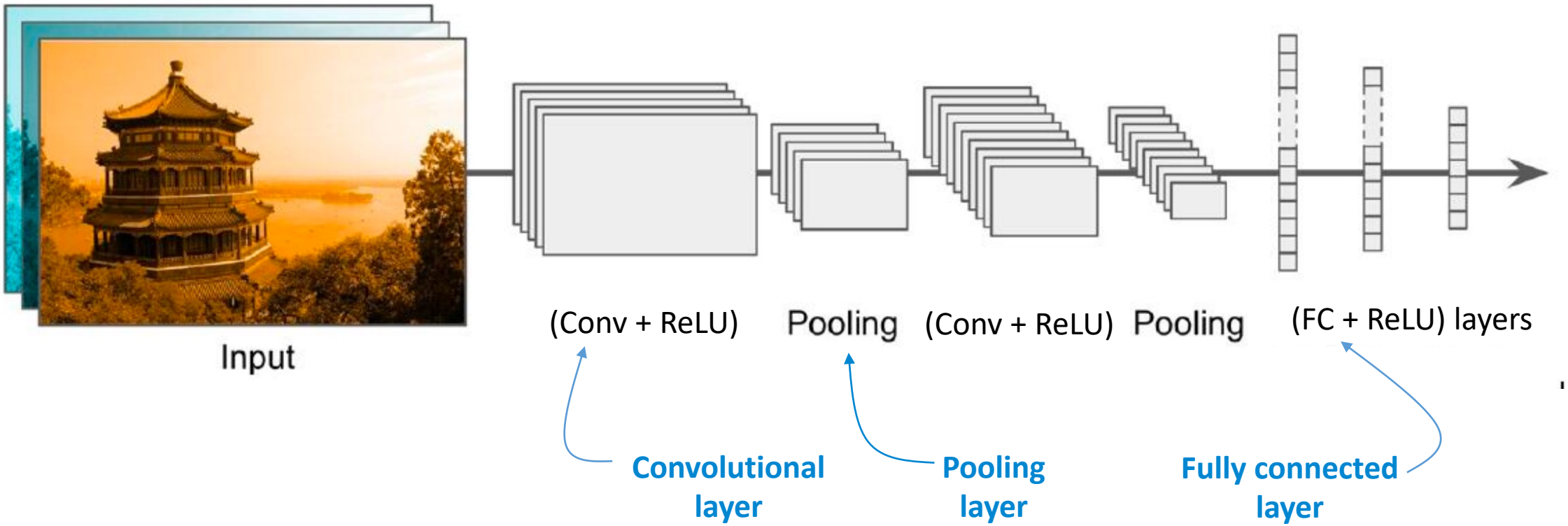
- **Convolution neural network**



- Detect local spatial patterns and extract them as feature maps by convolution
- Stack multiple convolutional layers to aggregate features hierarchically

Recap of Previous Session

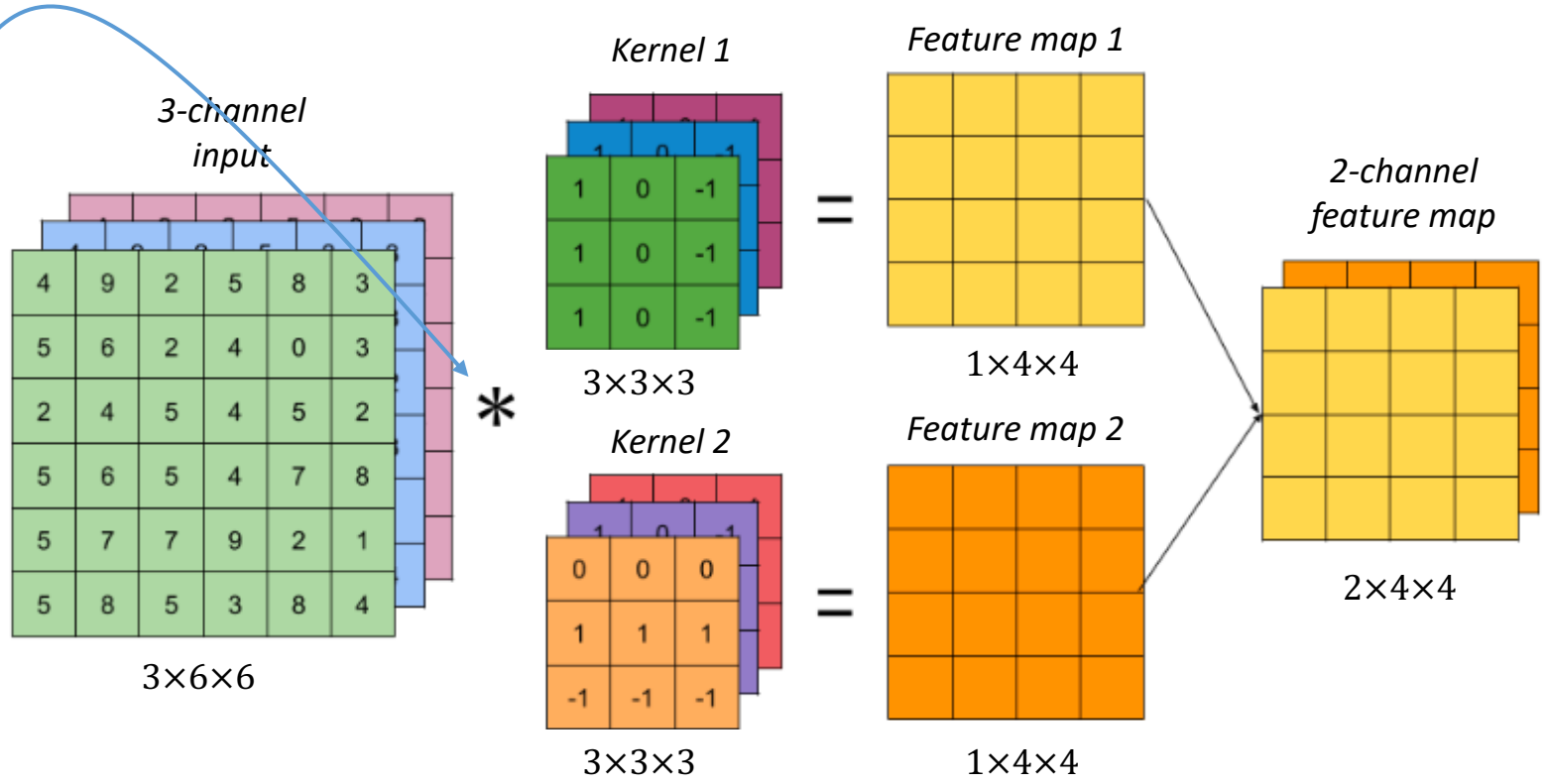
- **Typical CNN architecture**



Recap of Previous Session

- **Convolutional layer**
  - Multi-channel input image / feature map → Multi-channel output feature map

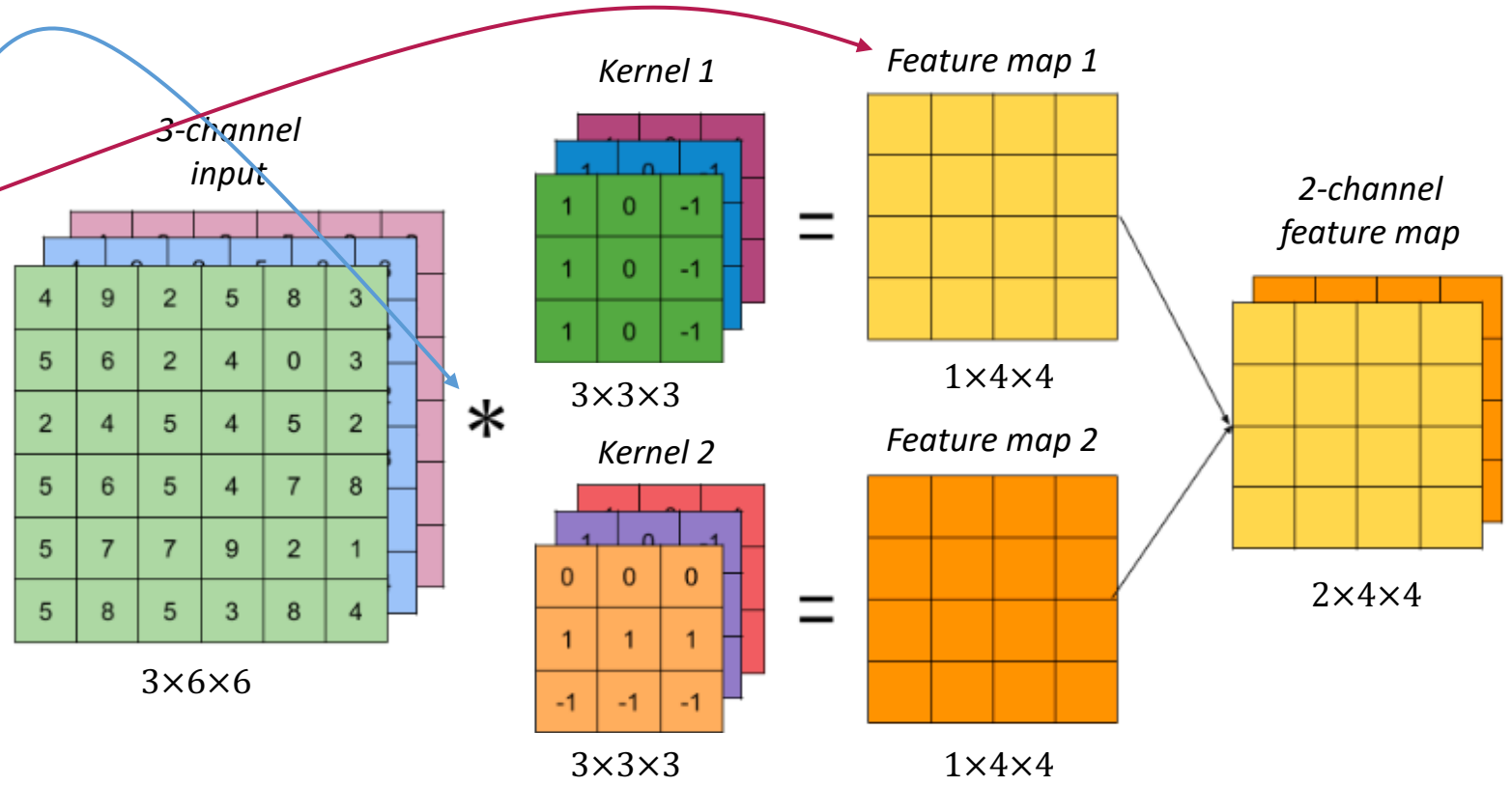
1. ***Convolution operation per input channel:***
   - ***Compute the sum of element-wise products in the kernel window***
   - ***Slide the kernel window across the input image / feature map***
     - ***From left to right***
     - ***From top to down***
   - ***Kernel size***
   - ***Padding***
   - ***Stride***



Recap of Previous Session

- **Convolutional layer**
  - Multi-channel input image / feature map → Multi-channel output feature map

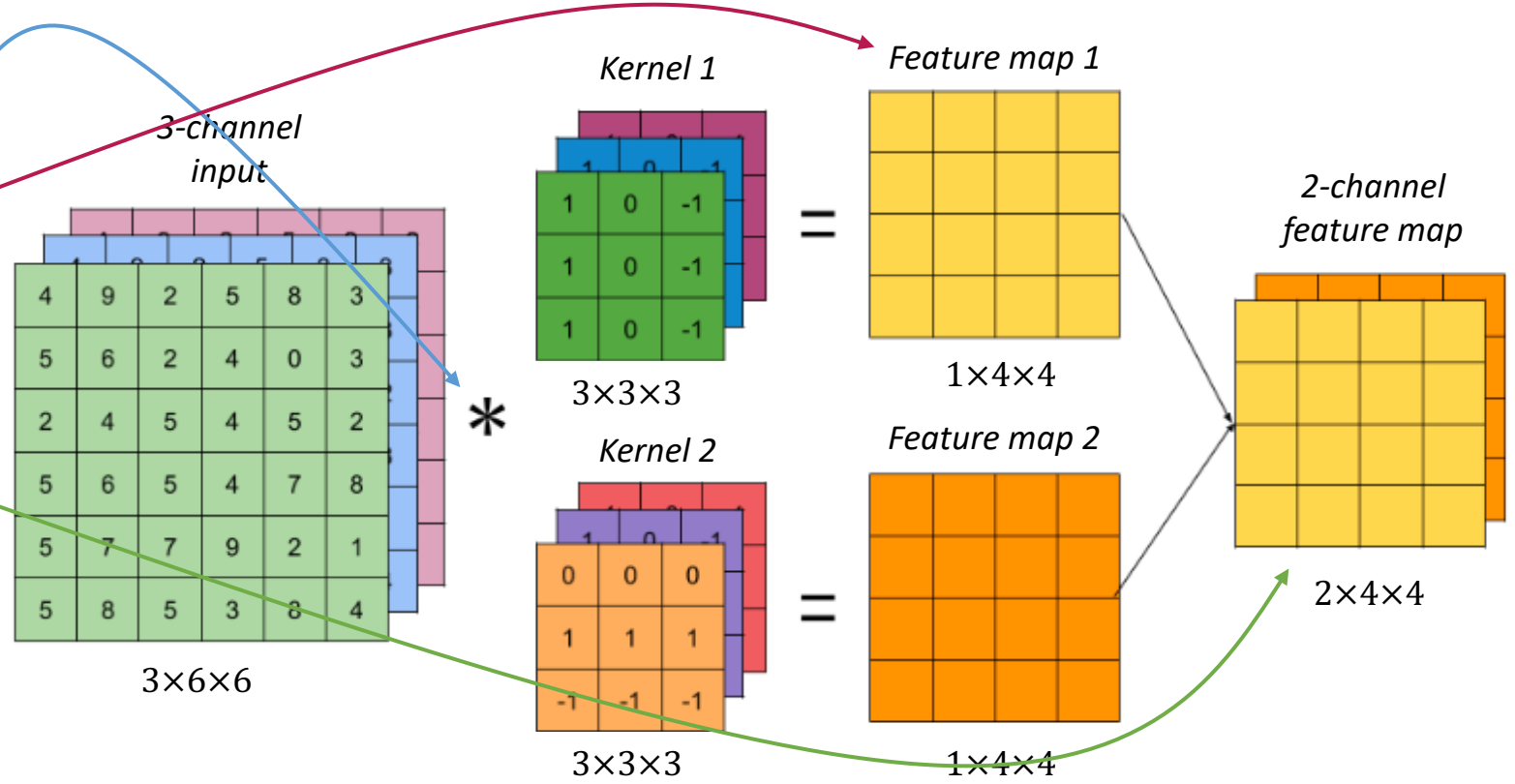
1. ***Convolution operation per input channel***
2. ***For multi-channel input, add up the channel-wise results to obtain the 1-channel output feature map***



Recap of Previous Session

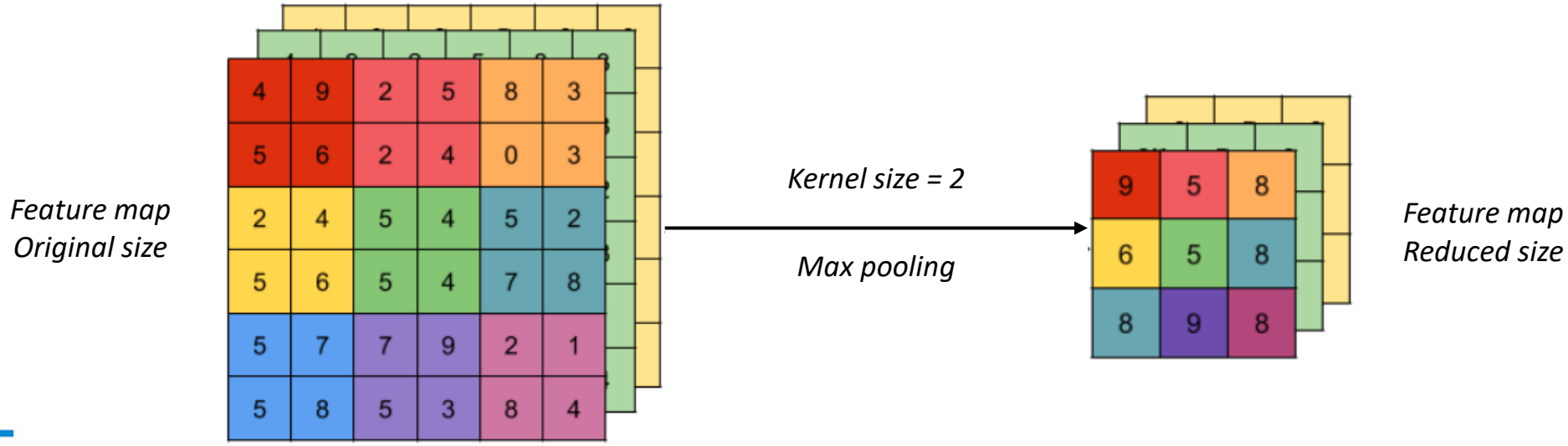
- **Convolutional layer**
  - Multi-channel input image / feature map → Multi-channel output feature map

1. *Convolution operation per input channel*
2. *For multi-channel input, add up the channel-wise results to obtain the 1-channel output feature map*
3. *To get multi-channel output, use multiple convolutional kernels, stack the resulting 1-channel output as the multi-channel output feature map*



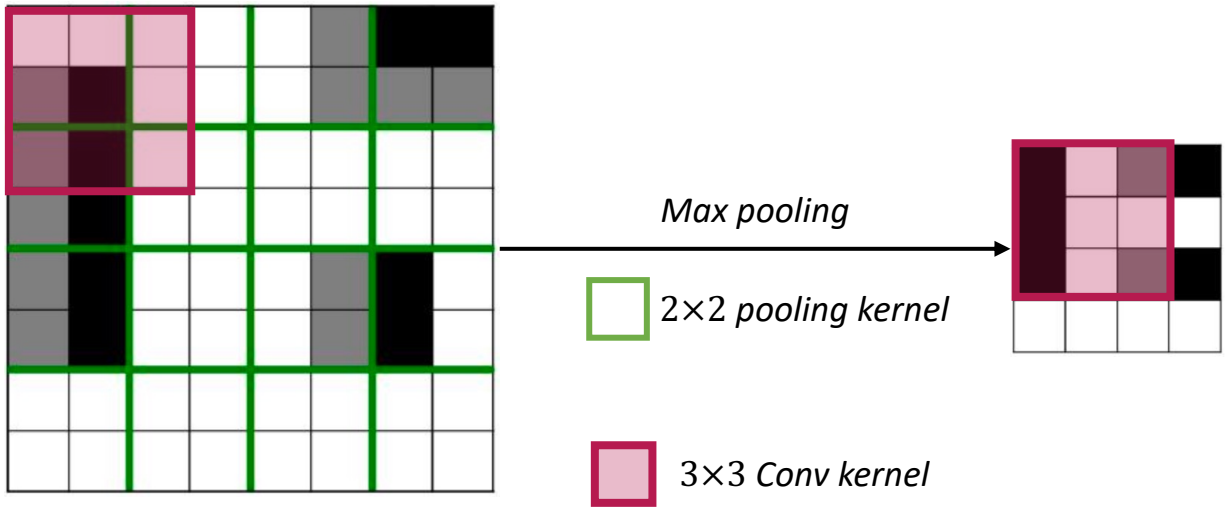
Recap of Previous Session

- **Pooling layer**
  - Reduce the size of feature map for extracting high-level features
  - **Max Pooling**
    - Output the maximum value in the kernel window
    - Slide the kernel window across the input feature map (no overlap by default)



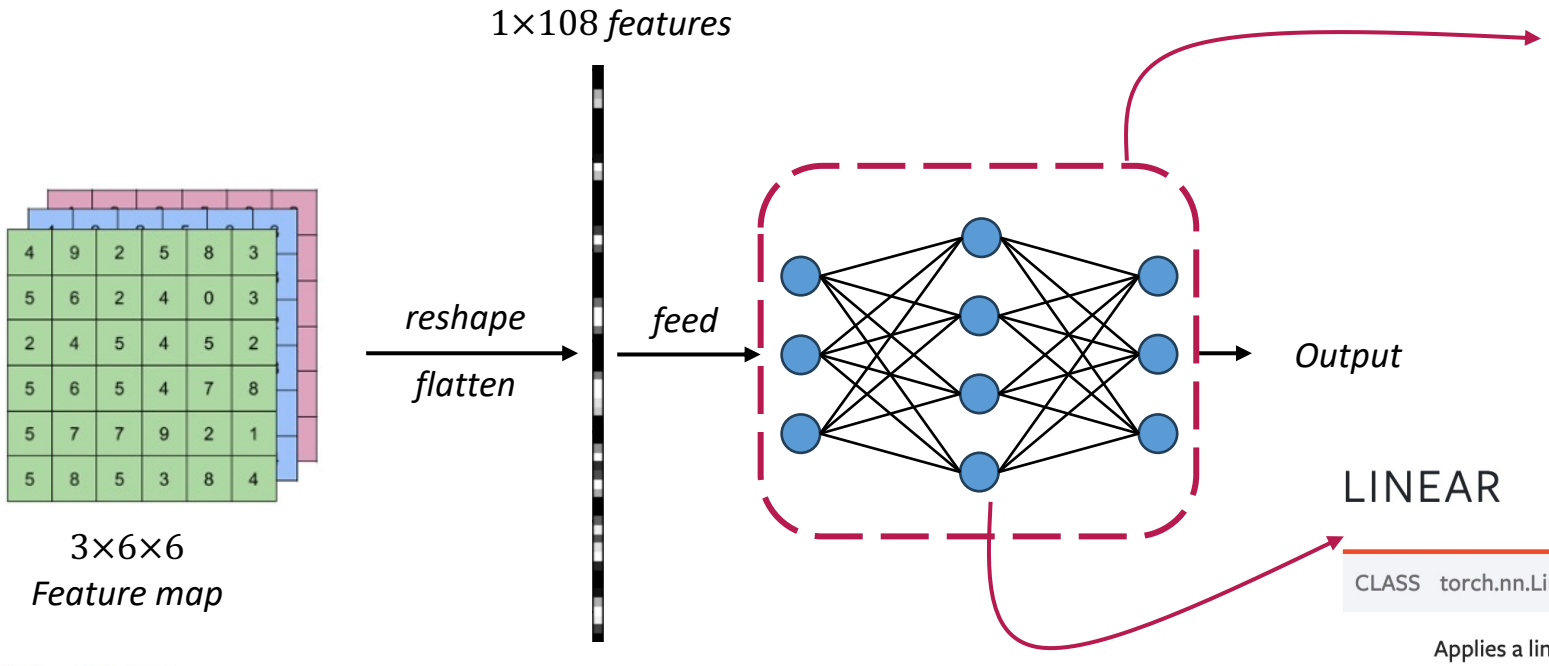
Recap of Previous Session

- **Pooling layer**
  - Reduce the size of feature map for extracting high-level features
  - **Max Pooling**
    - Preserve the global spatial pattern, ignore the local one
    - After pooling, the convolutional kernel can see a larger region to extract more global features



Recap of Previous Session

- **Fully connected layer**
  - Flatten the feature map as 1D feature vector
  - Use this 1D feature vector to make predictions



The part after flatten operation is a traditional artificial neural network.

- *Each layer is torch.nn.Linear.*
- *We refer to it as the fully connected (FC) layer in the CNN context.*
- *Each neuron in the FC layer is connected to all neurons in the previous layer.*

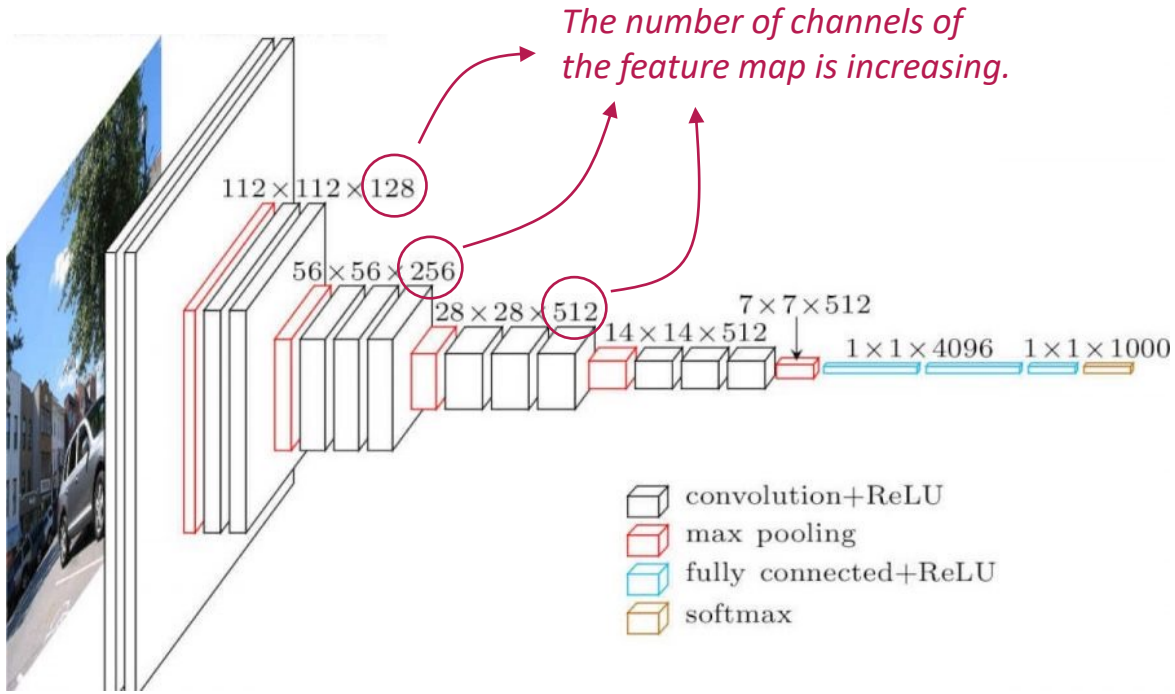
LINEAR

CLASS torch.nn.Linear(*in_features, out_features, bias=True, device=None, dtype=None*)

Applies a linear transformation to the incoming data: $y = xA^T + b$.

Recap of Previous Session

- **Pyramid architecture**:
  - As we go deeper in the CNN
    - **The size of feature map is shrinking** by convolution and pooling layers
      - Low-level feature → High-level feature
      - Local spatial pattern → Global spatial pattern
    - The number of channels of feature map is increasing
      - **Increase the number of convolutional kernels**
      - Corresponds to more possible combinations of simple features



Outline

- **Transfer learning**
- 1D Convolution neural network

Transfer learning

- Two cornerstones of deep learning
  - Deep neural networks
    - The deeper the model depth, the more learning parameters, the larger the capacity, and the stronger the representational power

Model	GPT-1	GPT-2	GPT-3	GPT-4
Parameter count	117 million	1.5 billion	175 billion	1.7 trillion

  - Large-scale training datasets
    - Small datasets are not enough to train complex neural networks
      - Small datasets are not representative enough
      - The network can easily over-fit to the small dataset
    - Large-scale datasets are more representative and contain more general knowledge
    - Models trained on large-scale general datasets can also perform well on specific tasks
      - For example, ChatGPT

Transfer learning

- Two cornerstones of deep learning
  - Deep neural networks
    - The deeper the model depth, the more learning parameters, the larger the capacity, and the stronger the representational power
  - Large-scale training datasets
    - Small datasets are not enough to train complex neural networks
      - Small datasets are not representative enough
      - The network can easily over-fit to the small dataset
    - Large-scale datasets are more representative and contain more general knowledge
    - Models trained on large-scale general datasets can also perform well on specific tasks
      - For example, ChatGPT

Training such networks can be very time-consuming.

Collecting and labeling such datasets are labor-intensive, and sometimes impossible for some specific tasks, for example, estimating the house price in a small village that only contains hundreds of houses.

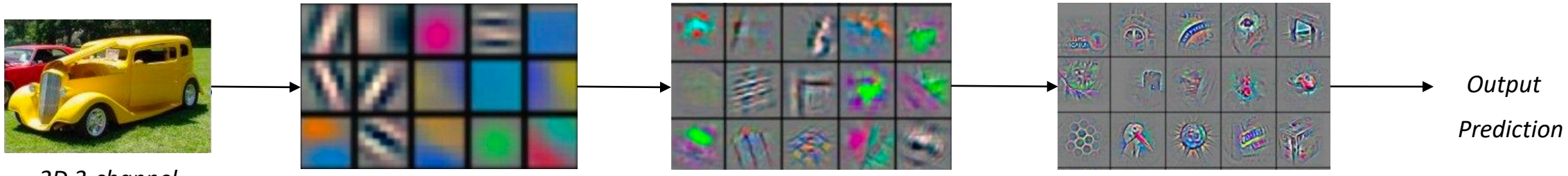
Can we re-use a deep neural network trained on a large-scale dataset and adapt it to a different task without requiring a full training process?

Transfer learning

- General definition of transfer learning:
  - A technique in machine learning in which knowledge learned from a task is re-used in order to boost performance on a related task
  - For example, knowledge gained while learning to recognize cars could be transferred to tasks such as recognizing trucks
- In the CNN context:
  - Transfer some knowledge from one neural network to another
  - Use a pre-trained model which has been trained on some large dataset
    - For example, ImageNet that contains more than 10 million images
  - Customized the pre-trained model to a specific task
    - Like distinguish between cat and dog

Transfer learning

- Rationale behind transfer learning
  - Domain-specific knowledge (like ML) is based on general knowledge regardless of domains (like mathematics, biology, statistics, etc.)

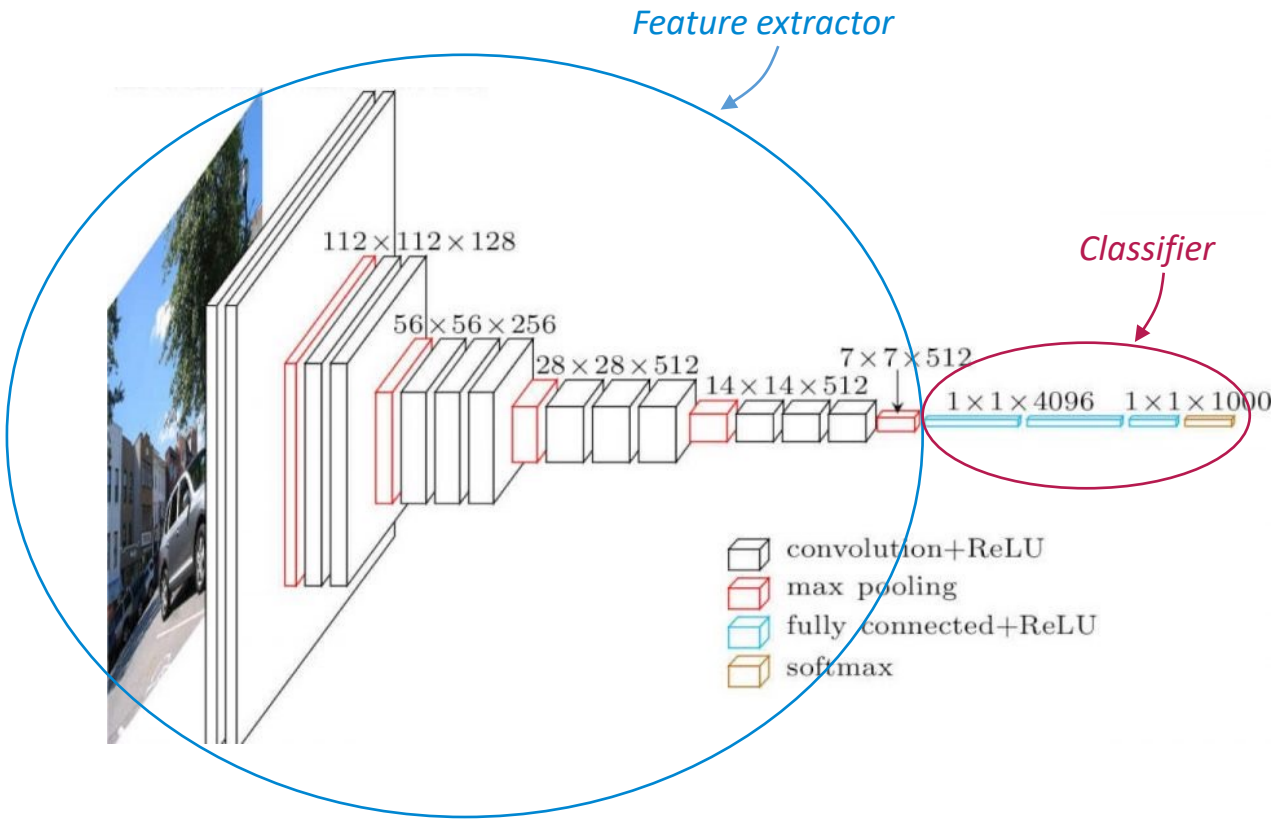


Different image classification tasks require the same or similar low-level or even mid-level features.

Deep CNNs trained on large-scale general image datasets can learn a better way to extract these features.

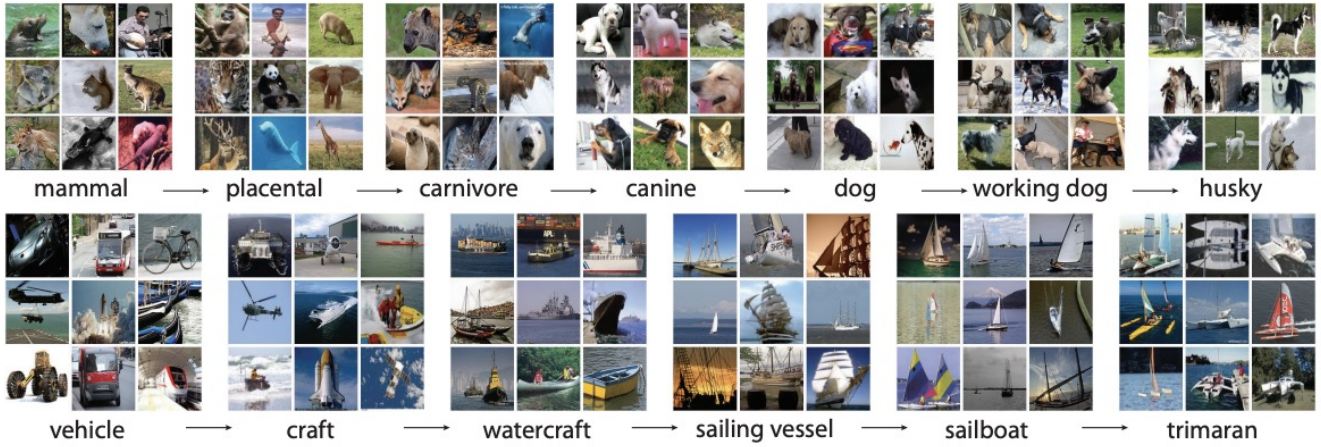
Transfer learning

- Rationale behind transfer learning
  - We can use the pre-trained CNNs to extract features and build a new classifier on top of these extracted features
    - Keep the feature extractor part unchanged
    - Re-build and re-train the classifier part
      - Only this classifier needs to be optimized, thus reduce the training time
    - (Optional) If we are not satisfied with the performance, we can then fine-tune the entire network with a small learning rate



Transfer learning

- Available resources
  - ImageNet
    - 1,281,167 training images
    - 50,000 validation images
    - 100,000 test images
    - 1,000 object classes
    - The object classes are based on the WordNet hierarchy
  - Pre-trained CNNs on ImageNet to predict the 1,000 classes
    - torchvision.models



Transfer learning

- Available resources
  - Pre-trained CNNs on ImageNet to predict the 1,000 classes
    - https://pytorch.org/vision/stable/models.html#table-of-all-available-classification-weights

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
AlexNet_Weights.IMAGENET1K_V1	56.522	79.066	61.1M	0.71	link
ConvNeXt_Base_Weights.IMAGENET1K_V1	84.062	96.87	88.6M	15.36	link
ConvNeXt_Large_Weights.IMAGENET1K_V1	84.414	96.976	197.8M	34.36	link
ConvNeXt_Small_Weights.IMAGENET1K_V1	83.616	96.65	50.2M	8.68	link
ConvNeXt_Tiny_Weights.IMAGENET1K_V1	82.52	96.146	28.6M	4.46	link
DenseNet121_Weights.IMAGENET1K_V1	74.434	91.972	8.0M	2.83	link
DenseNet161_Weights.IMAGENET1K_V1	77.138	93.56	28.7M	7.73	link
DenseNet169_Weights.IMAGENET1K_V1	75.6	92.806	14.1M	3.36	link
DenseNet201_Weights.IMAGENET1K_V1	76.896	93.37	20.0M	4.29	link
EfficientNet_B0_Weights.IMAGENET1K_V1	77.692	93.532	5.3M	0.39	link
EfficientNet_B1_Weights.IMAGENET1K_V1	78.642	94.186	7.8M	0.69	link
EfficientNet_B1_Weights.IMAGENET1K_V2	79.838	94.934	7.8M	0.69	link
EfficientNet_B2_Weights.IMAGENET1K_V1	80.608	95.31	9.1M	1.09	link
EfficientNet_B3_Weights.IMAGENET1K_V1	82.008	96.054	12.2M	1.83	link

- AlexNet
- ConvNeXt
- DenseNet
- EfficientNet
- EfficientNetV2
- GoogLeNet
- Inception V3
- MaxVit
- MNASNet
- MobileNet V2
- MobileNet V3
- RegNet
- ResNet
- ResNeXt
- ShuffleNet V2
- SqueezeNet
- SwinTransformer
- VGG
- VisionTransformer
- Wide ResNet

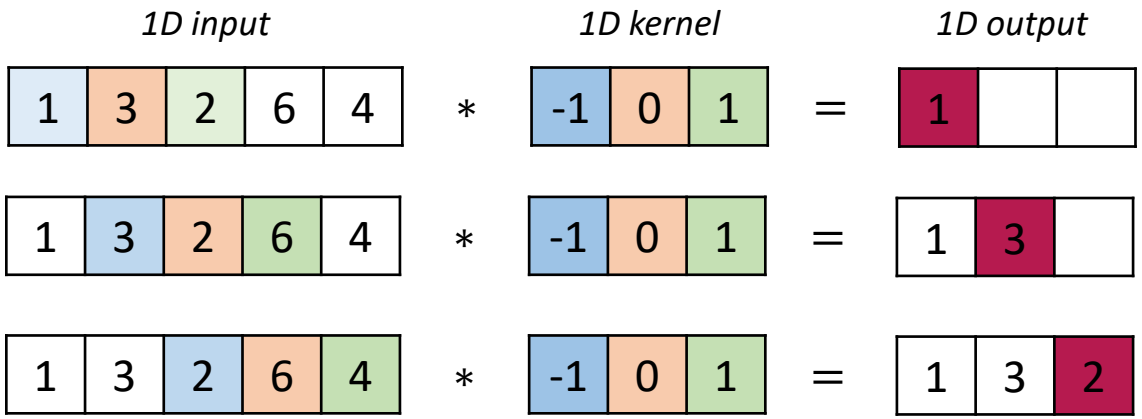
The simplest one

Outline

- Transfer learning
- **1D Convolution neural network**

1D Convolution neural network

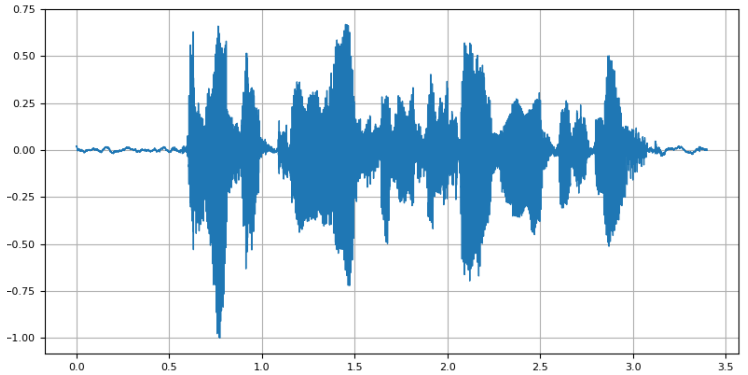
- Convolutional operation
  - Detect local spatial pattern / structure as feature
  - Not limited to 2D
- 1D Convolutional operation

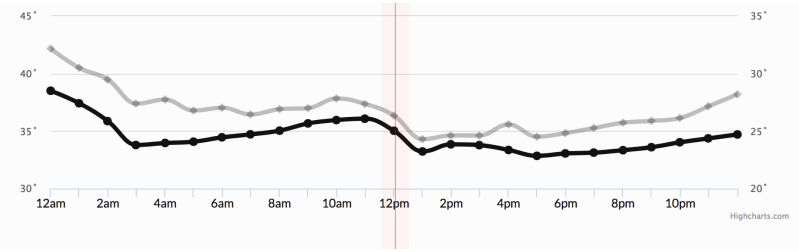


1D Convolution neural network

- Use traditional ANN when features are isolated
  - The order of features doesn't matter
  - For example, the flight price dataset
    - Airline name, source city, departure time, stops, …
- Use 1D CNN when the order of features matters
  - There are local spatial patterns between adjacent features

Audio data → 1D CNN for speech recognition





Time series data → 1D CNN for predicting temperature / stock price …

1D Convolution neural network

- The architecture of 1D CNNs is the same as the 2D ones
  - torch.nn.Conv2d → torch.nn.Conv1d
    - torch.nn.LazyConv2d → torch.nn.LazyConv1d
  - torch.nn.MaxPool2d → torch.nn.MaxPool1d
  - torch.nn.BatchNorm2d → torch.nn.BatchNorm1d
    - torch.nn.LazyBatchNorm2d → torch.nn.LazyBatchNorm1d

Hands-on Exercise

- **Exercise 07 CNN & Transfer Learning - Instruction**
- Exercise 07 CNN & Transfer Learning - Assignment