Lab 06. Deep Learning

Introduction to Computer Vision, Lab 06.

Today

- **Image classification**
- Recognition

Image classification

- Linear Model

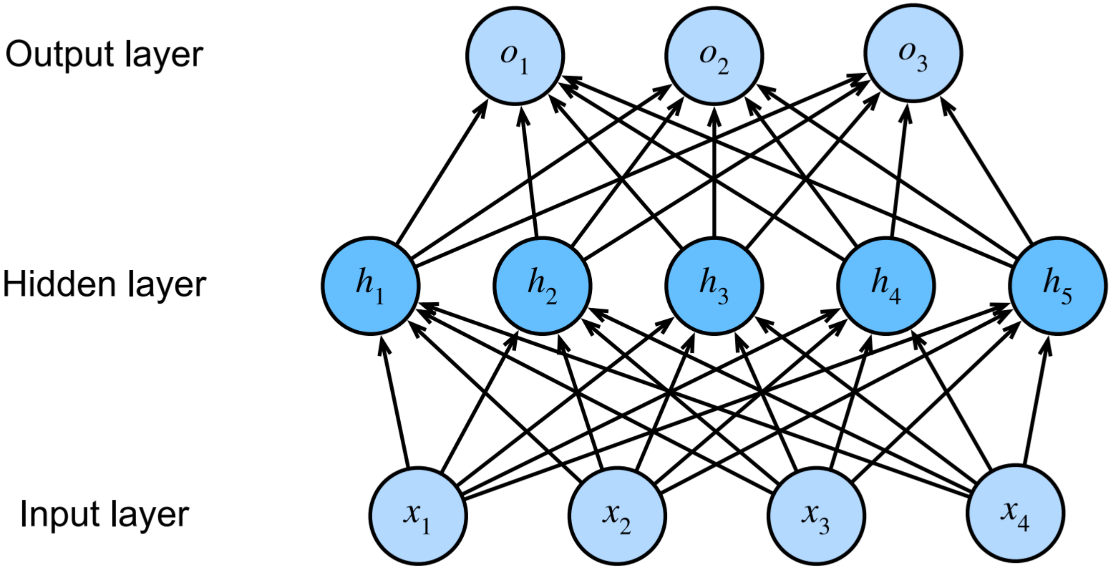
  - A linear model performs a single linear transformation and can only represent linear relationships.

    $y = Wx + b$

  - In image classification, this becomes a major limitation because **a linear model cannot capture the complex, nonlinear patterns present in images**.

Image classification

- Multi-Layer Perceptron
  - Multiple linear layers + nonlinear activation function.
  - MLP works well on small, simple image datasets, e.g., MNIST digits where spatial structure is less critical.



https://d2l.ai/d2l-en.pdf

Image classification

- Convolutional Neural Network
  - CNNs take advantage of the **locality** of image data by using small receptive fields that capture local patterns.
  - CNNs achieve **translation invariance** through weight sharing, allowing them to detect the same features regardless of where they appear in the image.

https://d2l.ai/d2l-en.pdf

Image classification

- Convolutional Neural Network
  - See interactive CNN demo: https://poloclub.github.io/cnn-explainer/

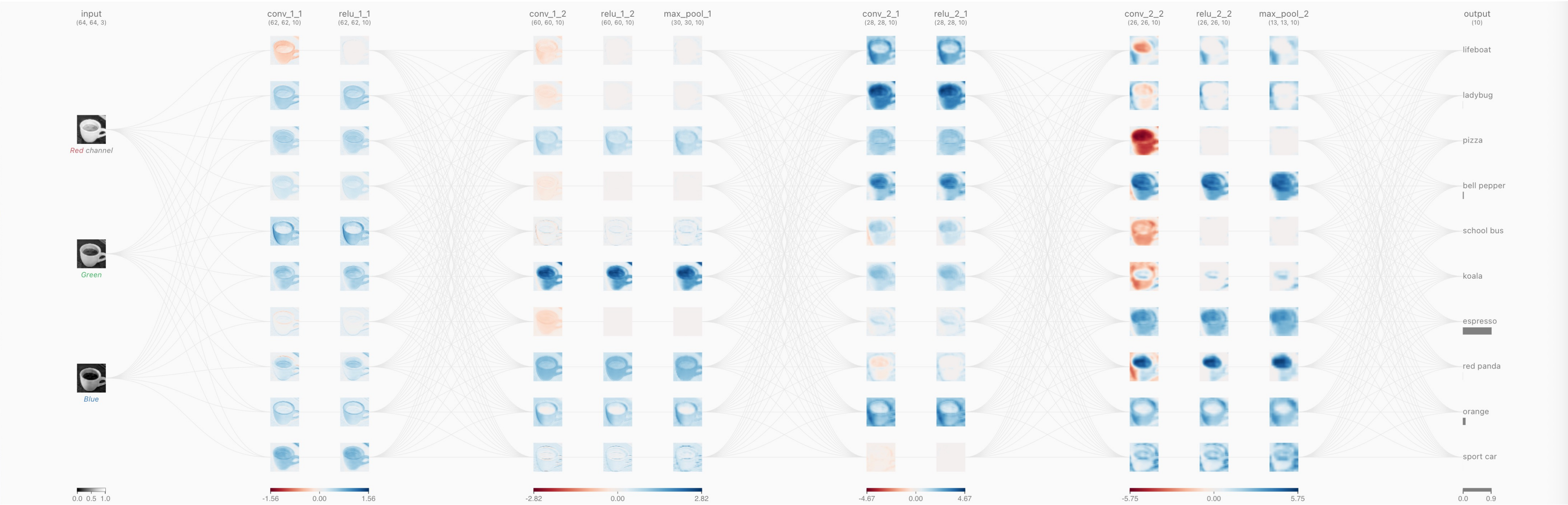


Image classification

- CIFAR10 dataset
  - The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.
  - There are 50000 training images and 10000 test images.
  - https://www.cs.toronto.edu/~kriz/cifar.html

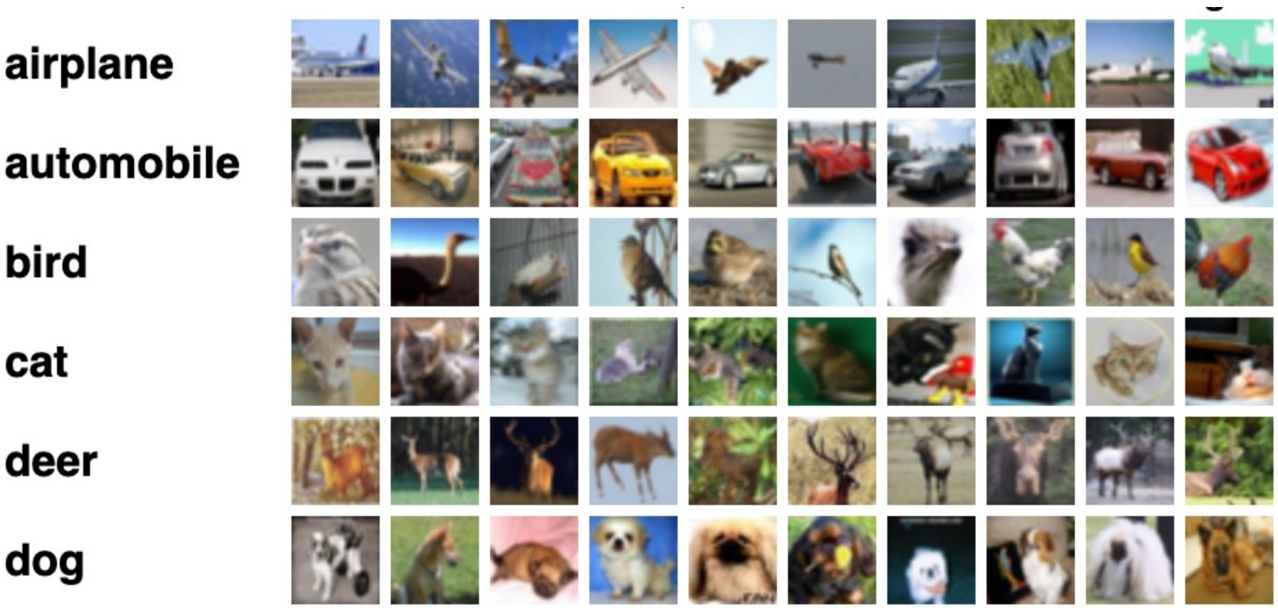


Image classification

- Please see classification.ipynb

- Train an MLP-based image classification model on CIFAR10 dataset.
- Build a CNN using PyTorch, train it and test it on CIFAR10 dataset.
- Add BN layers to the model, train it and test it on CIFAR10 dataset.

Pytorch Tutorial

- Please refer to slides from 2021 cs231n Lecture 06: 2021_cs231n_lec06.pdf

Today

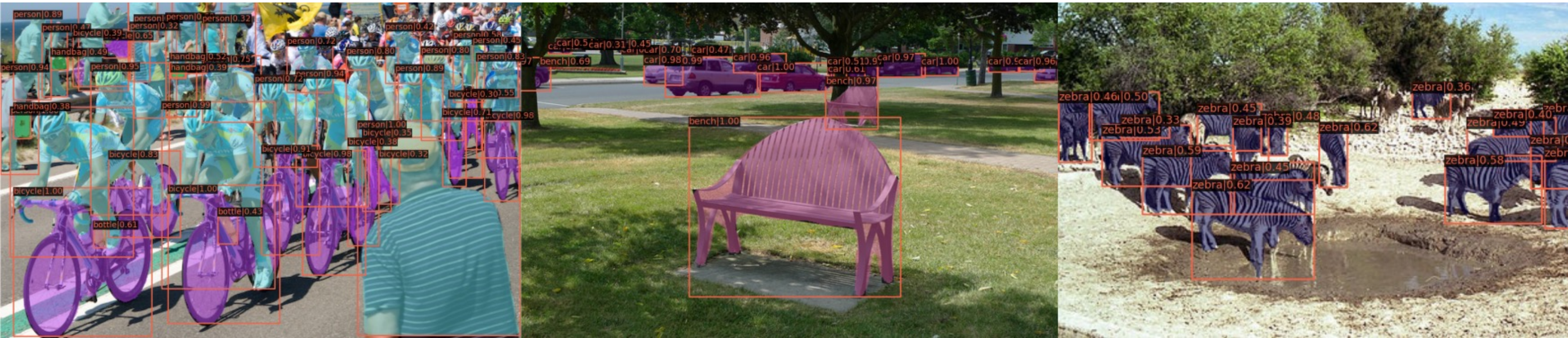
- Image classification
- **Recognition**

Recognition

- MMDetection

MMDetection is an open-source object detection and instance segmentation toolbox developed by OpenMMLab:

- **Modular design**: backbone, neck, head, loss, dataset, and pipelines can be freely combined.
- **Supports many classic and state-of-the-art models**, such as Faster R-CNN, YOLO series, RetinaNet, Mask R-CNN, and more.
- **Unified training/validation/inference workflow**.
- **Extensive model zoo** with many pretrained weights.



Recognition

- Popular 2D Detection Models
  - YOLOv3 / YOLOX
  - Fast RCNN / Faster RCNN
  - Mask RCNN / Cascade R-CNN

Recognition

- MMDetection Installation

- Latest:
  https://mmdetection.readthedocs.io/en/latest/get_started.html
- v2.20.0:
  https://docs.qq.com/slide/DR0FnSGpxSkp4bGhv

Tip: We provide two MMDet version examples of to accommodate different CUDA versions. You can choose **ANY MMDet version** which matches your local environment.

Recognition

- MMDetection Usage
  - Download the pretrained model

```
mim download mmdet --config rtmdet_tiny_8xb32-300e_coco --dest
checkpoints/
```

  - Run inference

```
from mmdet.apis import init_detector, inference_detector

config_file = 'configs_mmdet3.3/rtmdet/rtmdet_tiny_8xb32_300e_coco.py’
checkpoint_file = 'checkpoints/rtmdet_tiny_8xb32-
300e_coco_20220902_112414-78e30dcc.pth’

model = init_detector(config_file, checkpoint_file, device='cpu’)
img_path = 'pictures/det.jpg'
result = inference_detector(model, img_path)
```

Recognition

- Please see recognition_mmdet3.3.ipynb

- Running the MMDetection demo.
- Use your custom image.
- Use another model provided by this toolbox.

Submission

- Please only submit the two Jupyter notebooks.
- **Do not** submit any checkpoints or datasets.