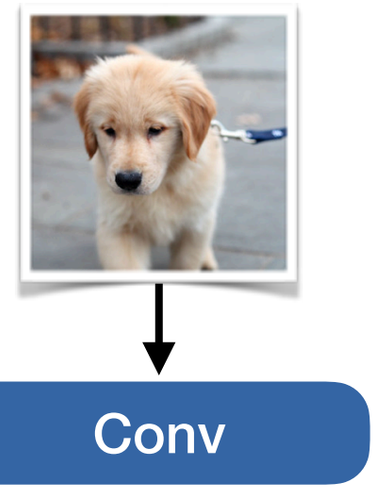
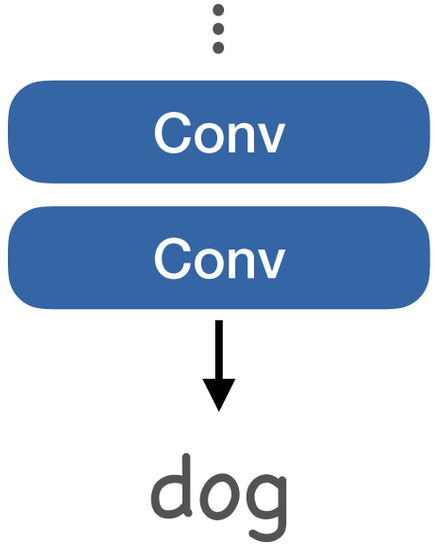
Course summary and further topics

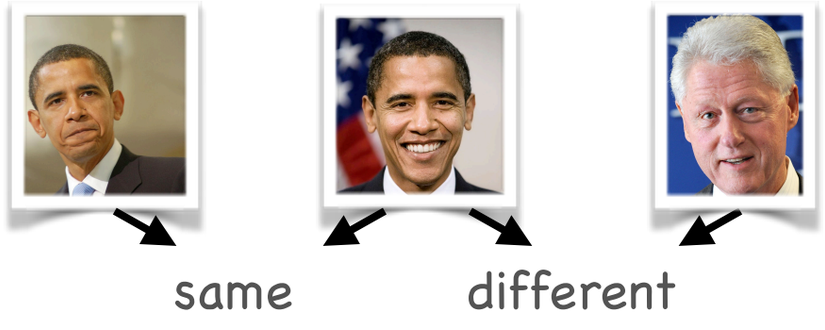
This course

- Fundamentals
  - How to build, train, use deep networks
- Few applications

```
In [ ]: %pylab inline
        import torch
        import sys
        sys.path.append('..')
        sys.path.append('../..')
        from data import load
        train_data, train_label = load.get_dogs_and_cats_data(resize=(128,128), n_images=10)
        device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
        print('device = ', device)

In [ ]: class ConvNet(torch.nn.Module):
            class Block(torch.nn.Module):
                def __init__(self, n_input, n_output, stride=1):
                    super().__init__()
                    self.net = torch.nn.Sequential(
                      torch.nn.Conv2d(n_input, n_output, kernel_size=3, padding=1, stride=stride),
                      torch.nn.ReLU(),
                      torch.nn.Conv2d(n_output, n_output, kernel_size=3, padding=1),
                      torch.nn.ReLU()
                    )

                def forward(self, x):
                    return self.net(x)

            def __init__(self, layers=[32,64,128], n_input_channels=3):
                super().__init__()
                L = [torch.nn.Conv2d(n_input_channels, 32, kernel_size=7, padding=3, stride=2),
                     torch.nn.ReLU(),
                     torch.nn.MaxPool2d(kernel_size=3, stride=2, padding=1)]
                c = 32
                for l in layers:
                    L.append(self.Block(c, l, stride=2))
                    c = l
                self.network = torch.nn.Sequential(*L)
                self.classifier = torch.nn.Linear(c, 1)

            def forward(self, x):
                # Compute the features
                z = self.network(x)
                # Global average pooling
                z = z.mean(dim=[2,3])
                # Classify
                return self.classifier(z)[:,0]

        net = ConvNet()
```

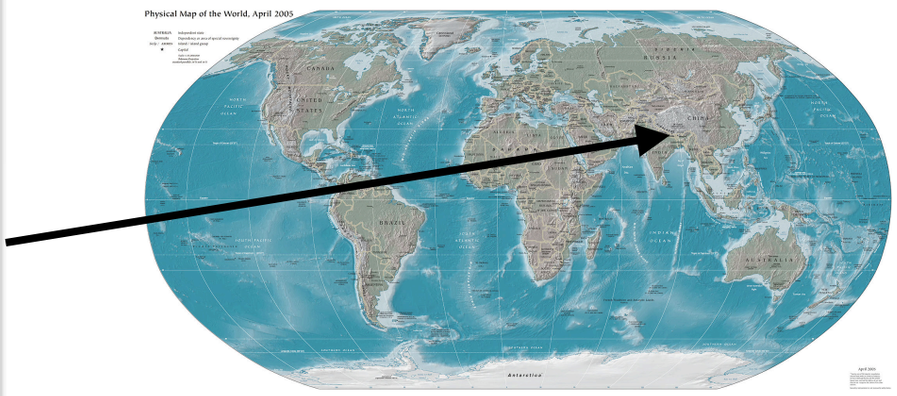


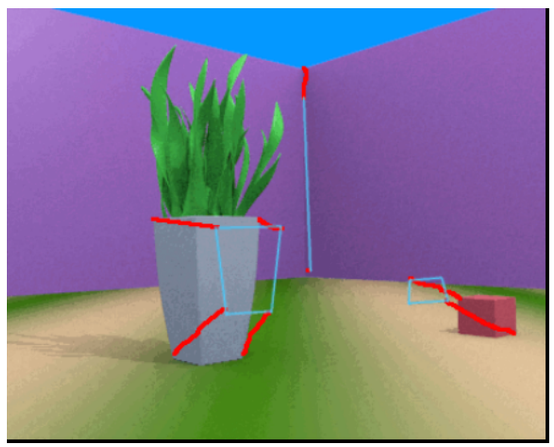




Computer vision

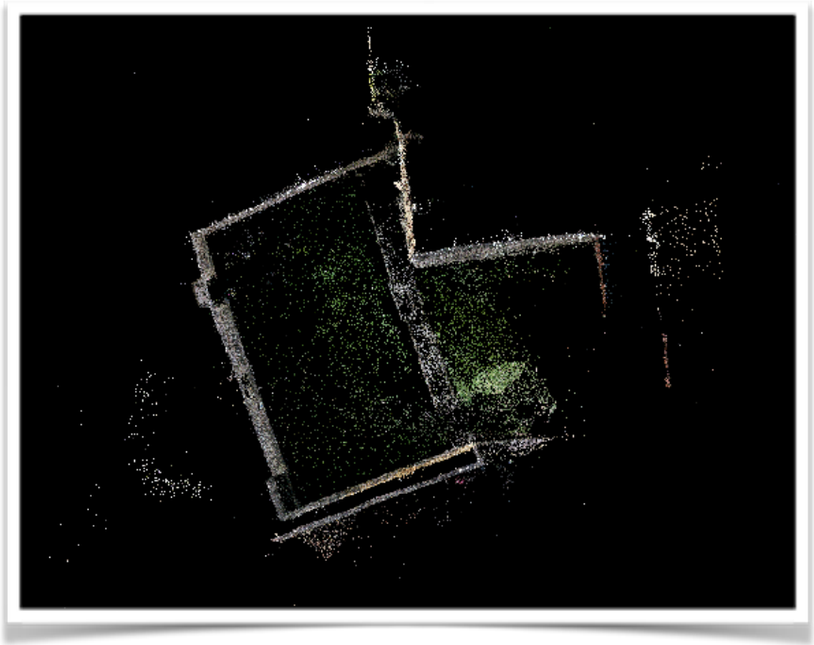
- Geolocalization
- Pose estimation
- Tracking
- Scene layout estimation
- Visual odometry
- …

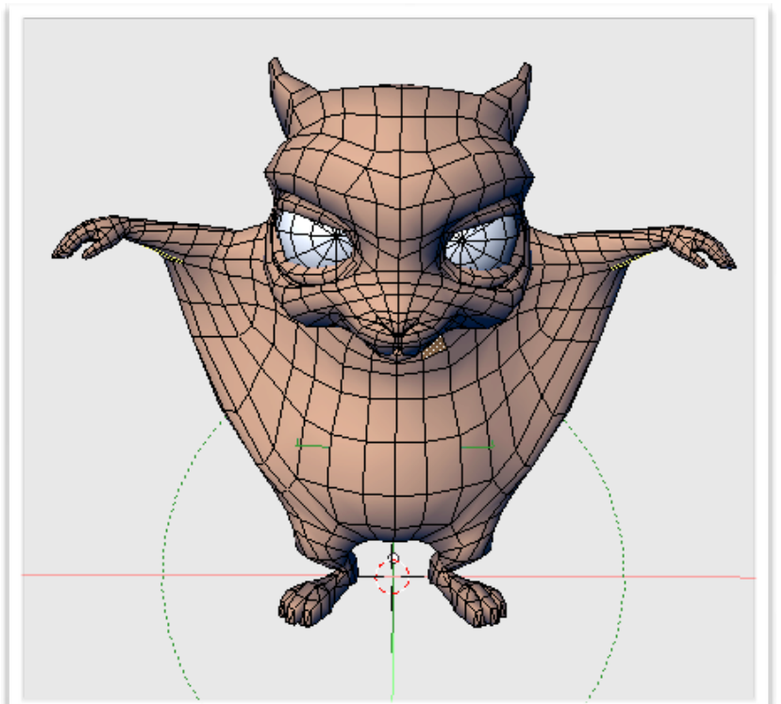




3D vision

- Point cloud or volume based networks
- Applications
- Reconstruction
- 3D recognition
- Surface representation
- …



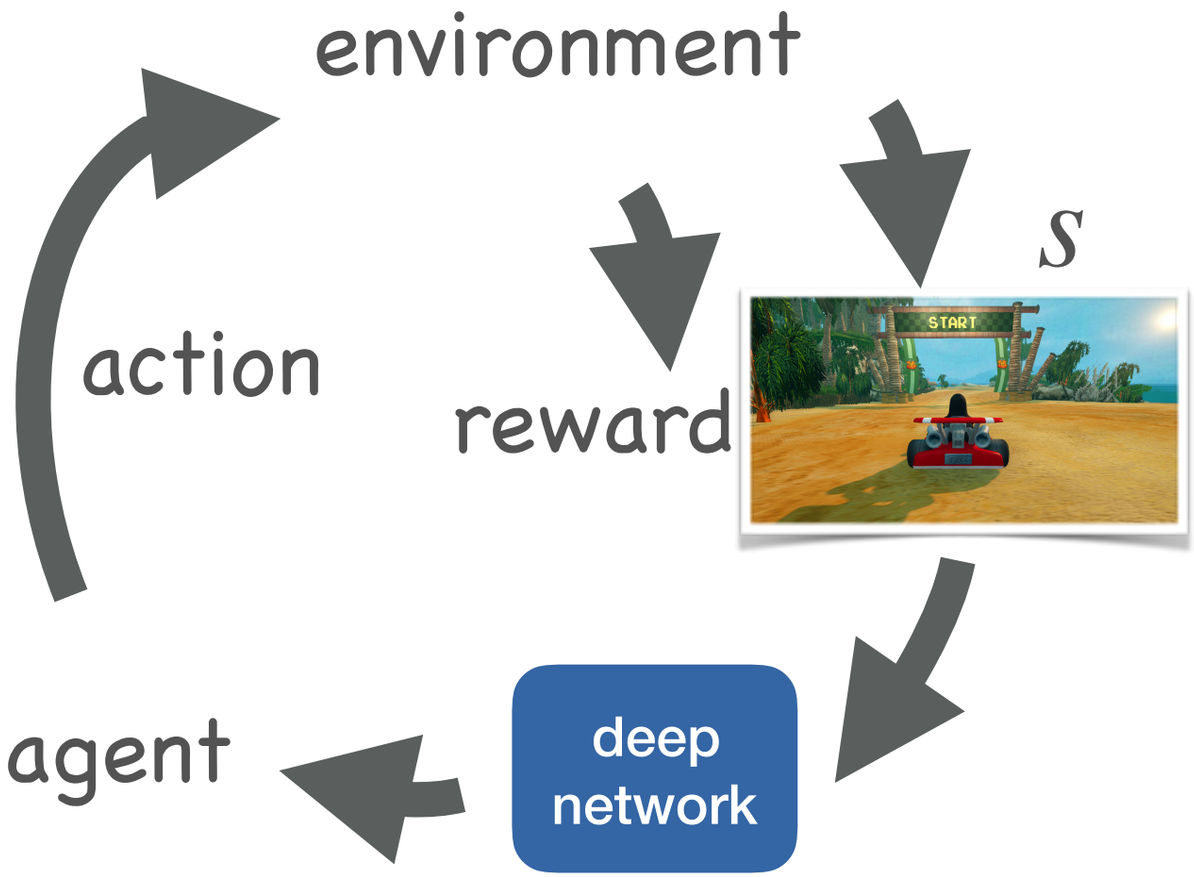


Natural language processing

- Word based models
- Applications
- Translation
- Sentiment analysis
- Topic modelling
- …

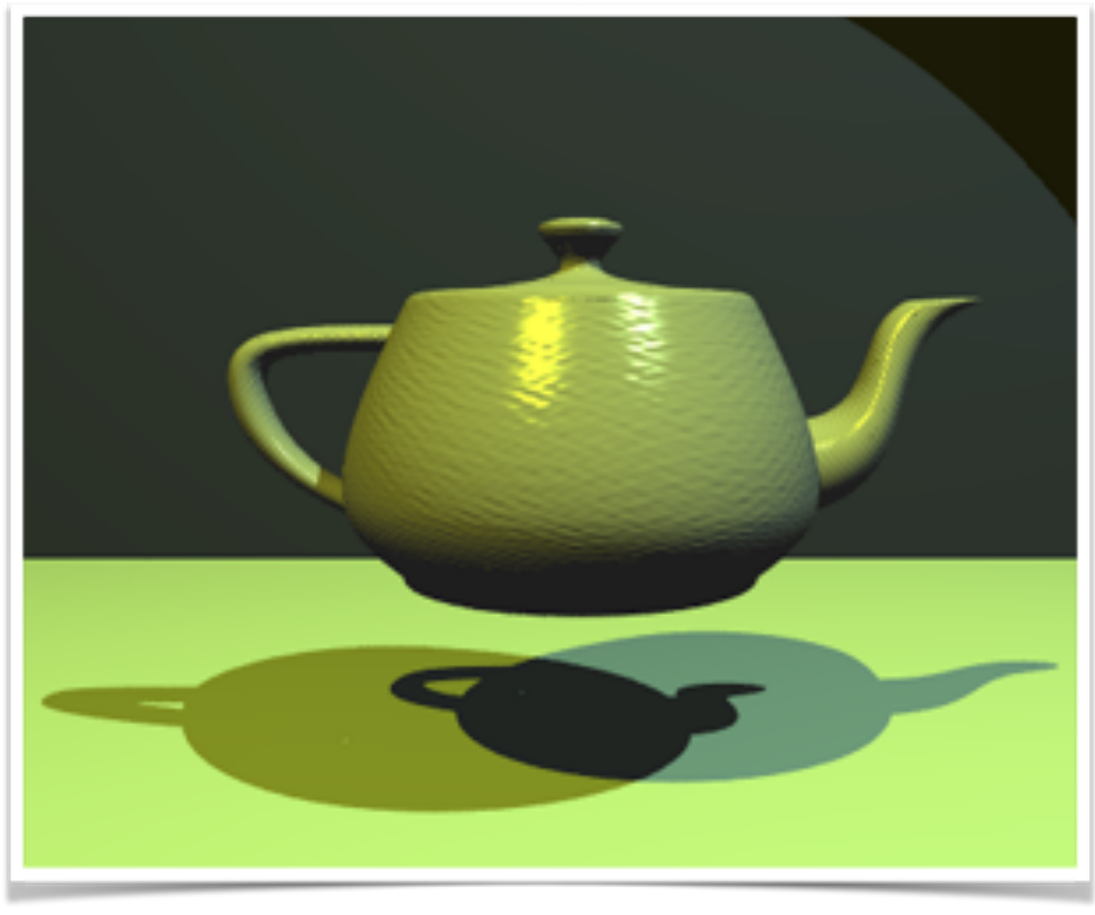
Reinforcement learning

- Q-learning
- Policy gradient++
- Applications
  - Robotics
  - Meta-learning
  - …



Compute graphics

- Generative models
- Applications
  - Matting
  - Image editing
  - Physical simulation
  - …



Deep learning hardware and architecture

- How do we implement any of this efficiently?
  - Fast matrix multiplications
  - Hardware support
  - …