2020.11.6-11.周报

本周对李宏毅老师机器学习课程中的Network Compression和Seq2seq进行复习总结，并完成作业六Adversarial Attack的实验。

一、网络压缩(NETWORK COMPRESSION)

1. Network Pruning

- importance of a weight:

$$L1, L2, \dots$$

通过这两种正则化方法来判断某一个变量是否重要。

- After pruning,the accracy will drop.
- Fine-tuning on training data for server.
- Don't prune too much at once,or the network won't recover.
- Practical Issue:

  * Weight pruning:The network architecture becomes irregular.(剪裁网络中的权重，深度网络会变得不规律，在实现的时候很难通过GPU加速)

  *Neuron pruning:The network architecture is regular.(剪裁对应的神经元，深度网络依然是规律的，方便GPU加速)

2. Knowledge Distillation

假设我们以及有一个训练好的完整的深度神经网络，但是这样的网络在一些设备上无法部署，所以可以设计一个规模较小的神经网络，然后用大的网络(Teacher Net)去训练小的网络(Student Net).在训练的时候有一点需要注意，在原始的网络中做sigmoid以及sofmax后，概率较小的分类，它们最后得到的y值会同样小，这样小的差距包含的信息太少，不足以去训练小的神经网络，所以需要我们给出另一种计算方法：

$$y_i = \frac{exp(x_i/T)}{\sum_j exp(x_i/T)}$$

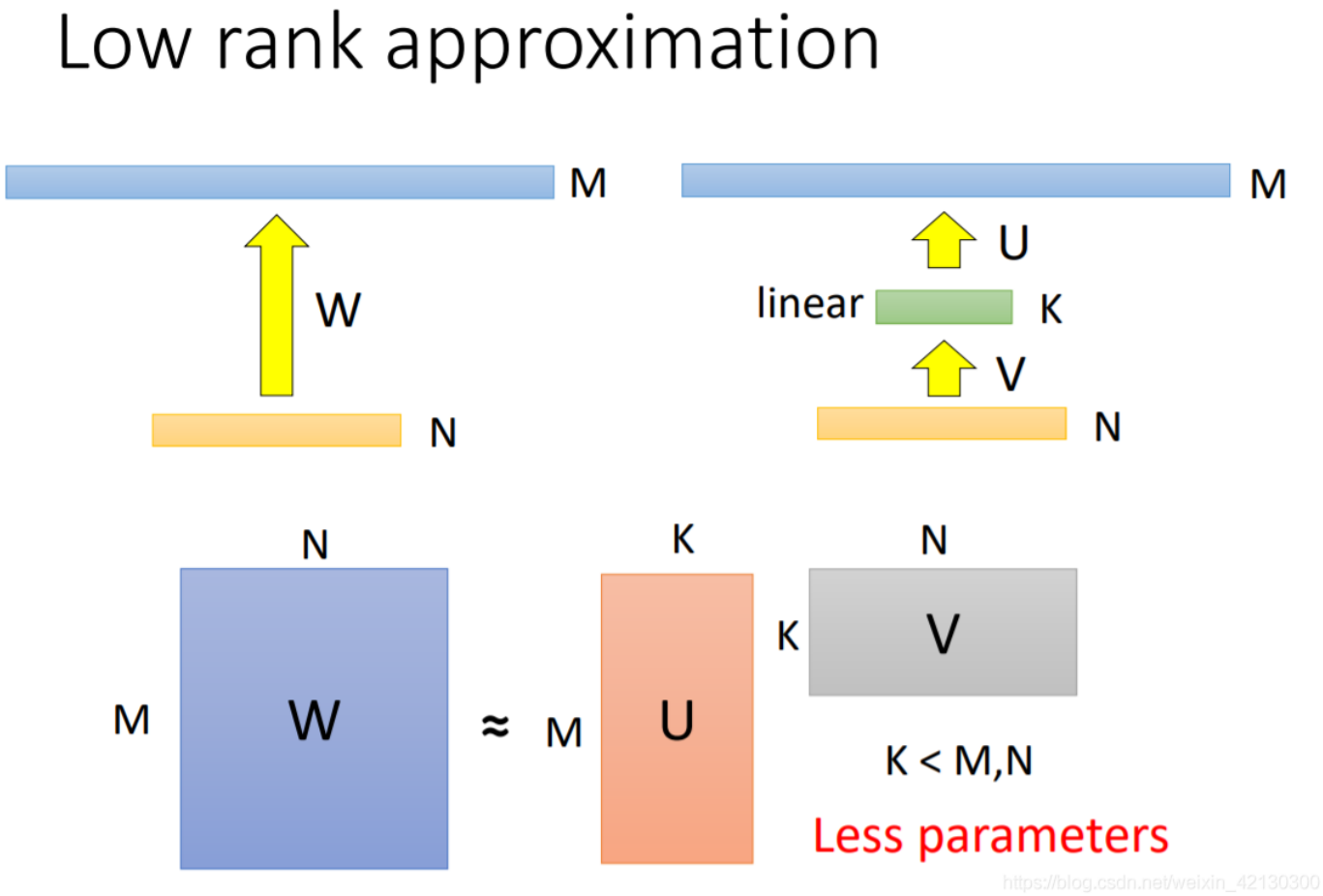
这里假设T是x_i中最大的那个数，假设有$x_1 = 100, x_2 = 10, x_3 = 1$，在同时除以$T$后，得到的$y$值分别为:$y_1 = 0.56, y_2 = 0.23, y_3 = 0.21$，可以发现，这样得到的$y_2, y_3$差距就比较小，可以用来去训练小的神经网络模型。

3. Parameter Quantization

- Using less bits to represent a value.
- Weight clustering
- Represent frequent clusters by less bits,represent rare clusters by more bits
  e.g.Huffman encoding

4. Architecture Design

- Low rank approximation

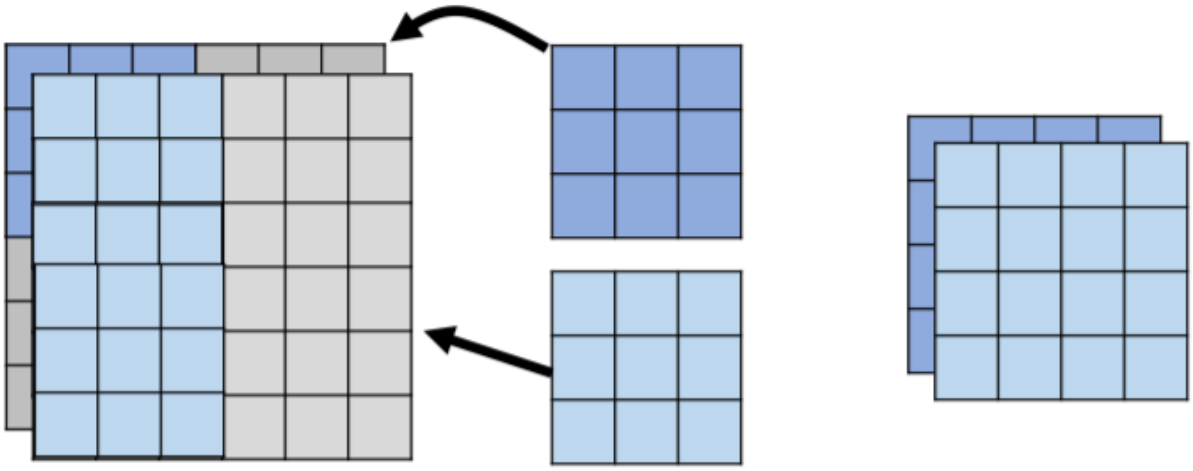


在原始问题中，我们在神经网络的输入特征数为N,经过W矩阵的运算后，得到了M；为了减少原来计算过程中过多的参数，我们可以对矩阵W进行线性分解，分解成U，V这两个矩阵的乘积，并且：$I:number\ of\ input\ channels$；$O:number\ of\ output\ channels$；$k*k:kernal\ size$.下面给出一种新的计算方法：

- Depthwise Separable Convolution

  这种算法想到的是，首先是设计卷积核的数量：Filter number = Input channel number。如下图所示，假设我们给出的是一种灰度图像，那么Filter number = 2。在计算的过程中，每个Filter只考虑自己所需要的做卷积的那一层就好。做完后再使用 Pointwise Convolution做卷积，这一步和传统的CNN卷积神经网络的做法一致。

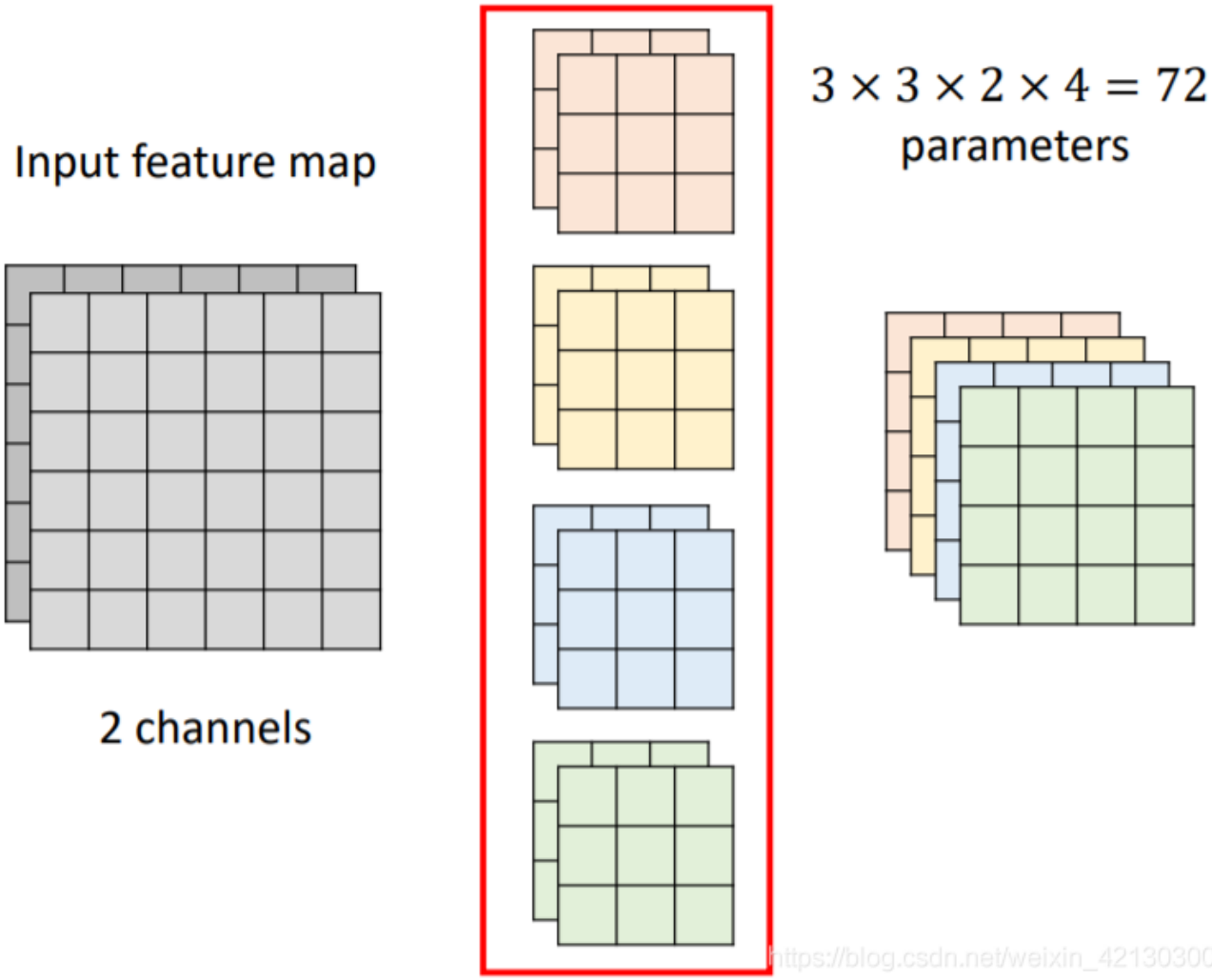
1. Depthwise Convolution



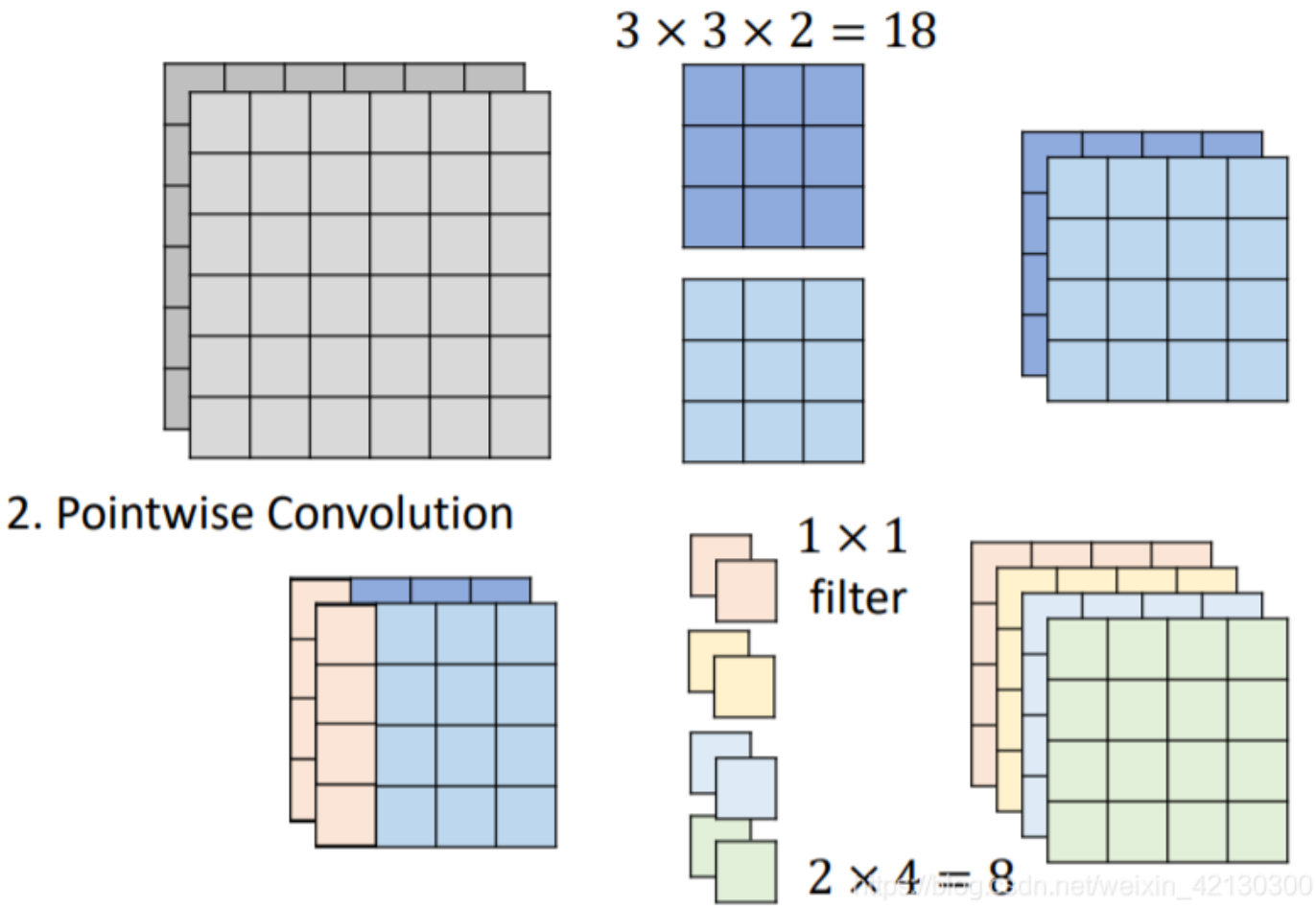
- Filter number = Input channel number
- Each filter only considers one channel.
- The filters are $k \times k$ matrices
- There is no interaction between channels.

回顾传统的CNN的做法，我们可以得到的中间参数为:$3 * 3 * 2 * 4 = 72$:

Review: Standard CNN



但是Depthwise Separable Convolution只需要$3 * 3 * 2 + 2 * 4 = 26$的参数就好。



最后得到的结果如下：

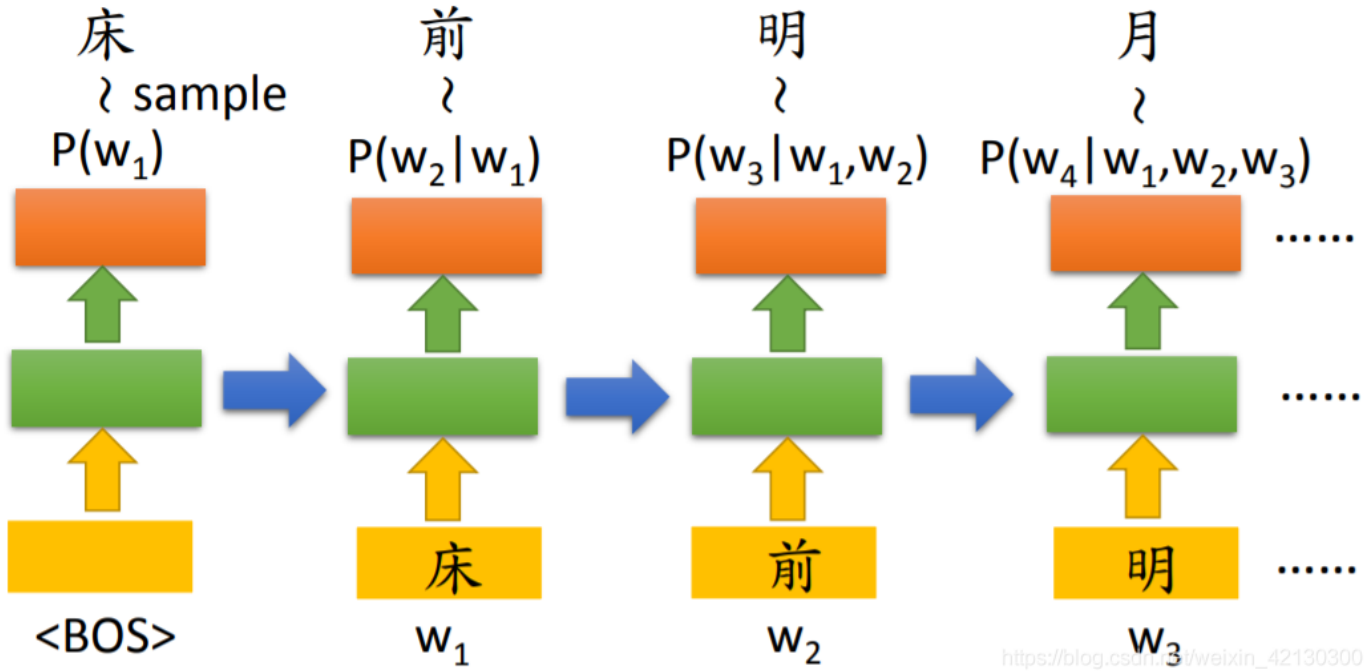
$$\frac{K*K*I+I*O}{K*K*I*O}=\frac{1}{O}+\frac{1}{K*K}$$

二、Seq2seq

1. Conditional Generationby RNN & Attention

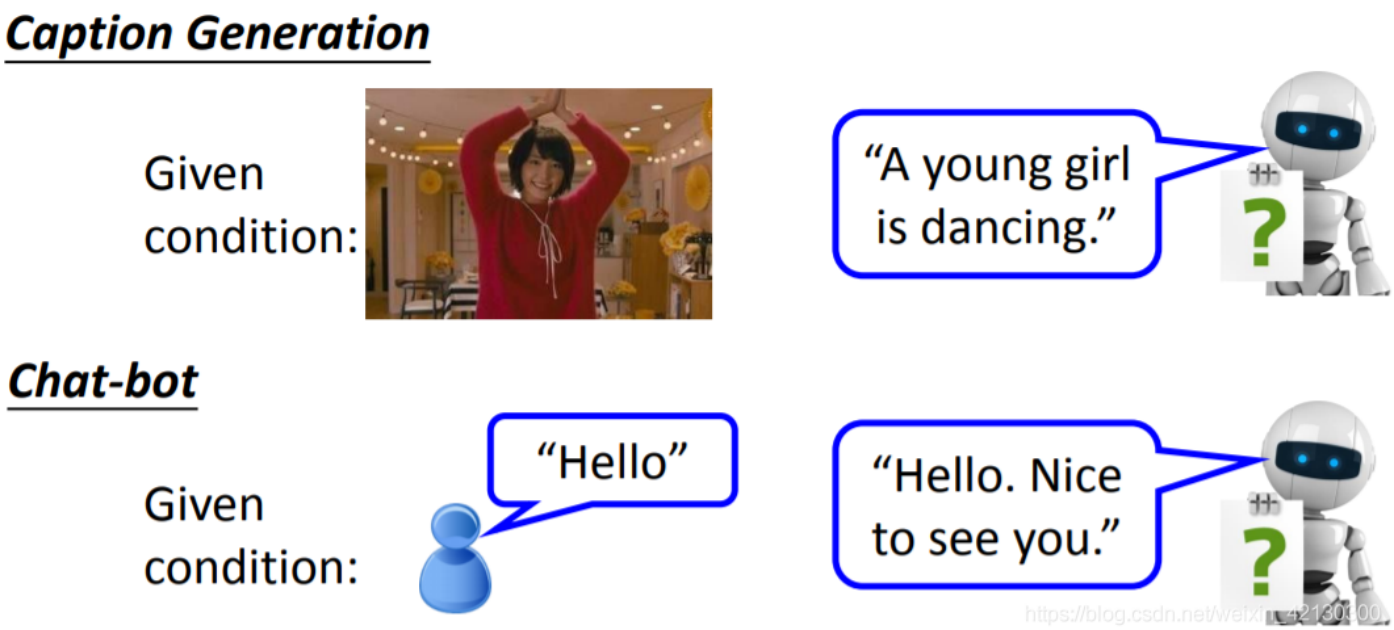
- Generation

  如果我有一个输入序列，比如“床前明月光”，模型在接受到这句话的开始时，会自动输出“床”，输入“床”后，模型可以继续输出“前”，以此类推。



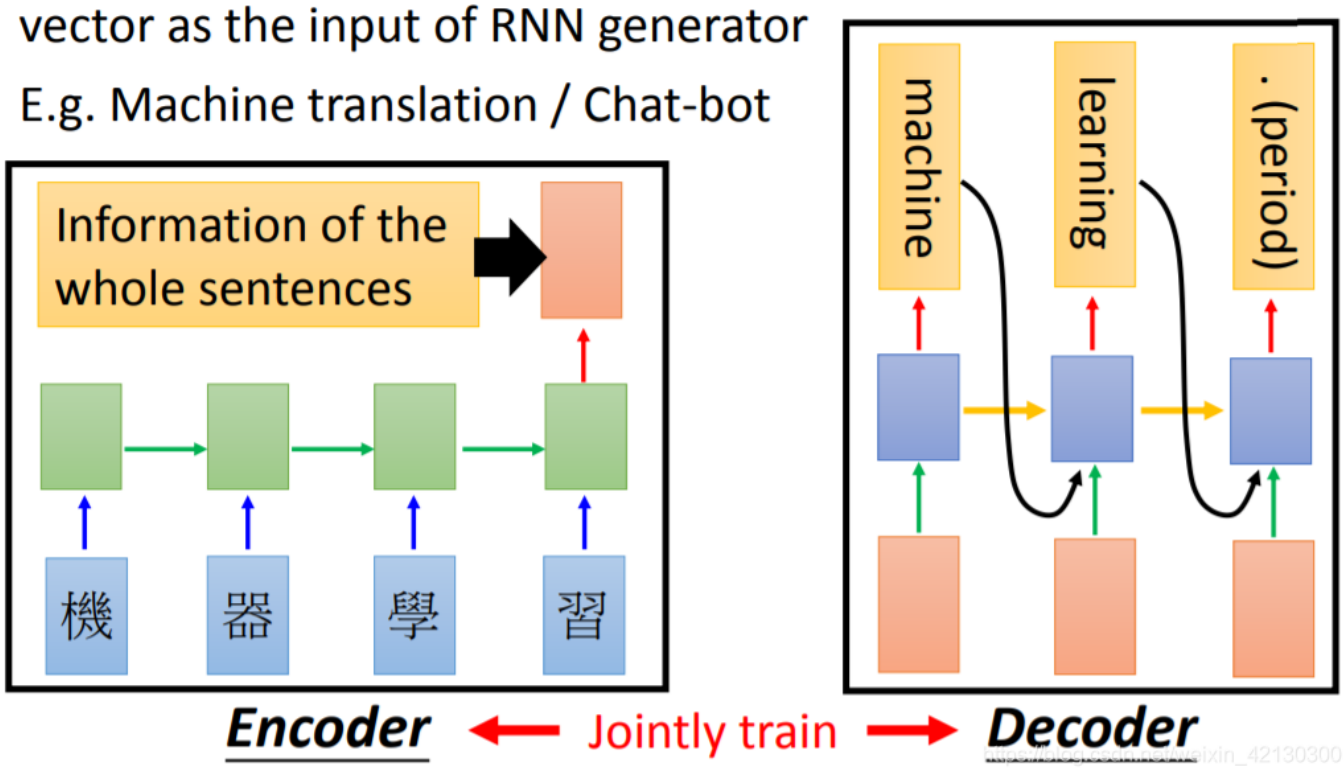
- Conditional Generation(Sequence-to-sequence learning)

  一般情况下，我们希望机器并不是简单地输出，而是可以根据条件，输出特定的内容：

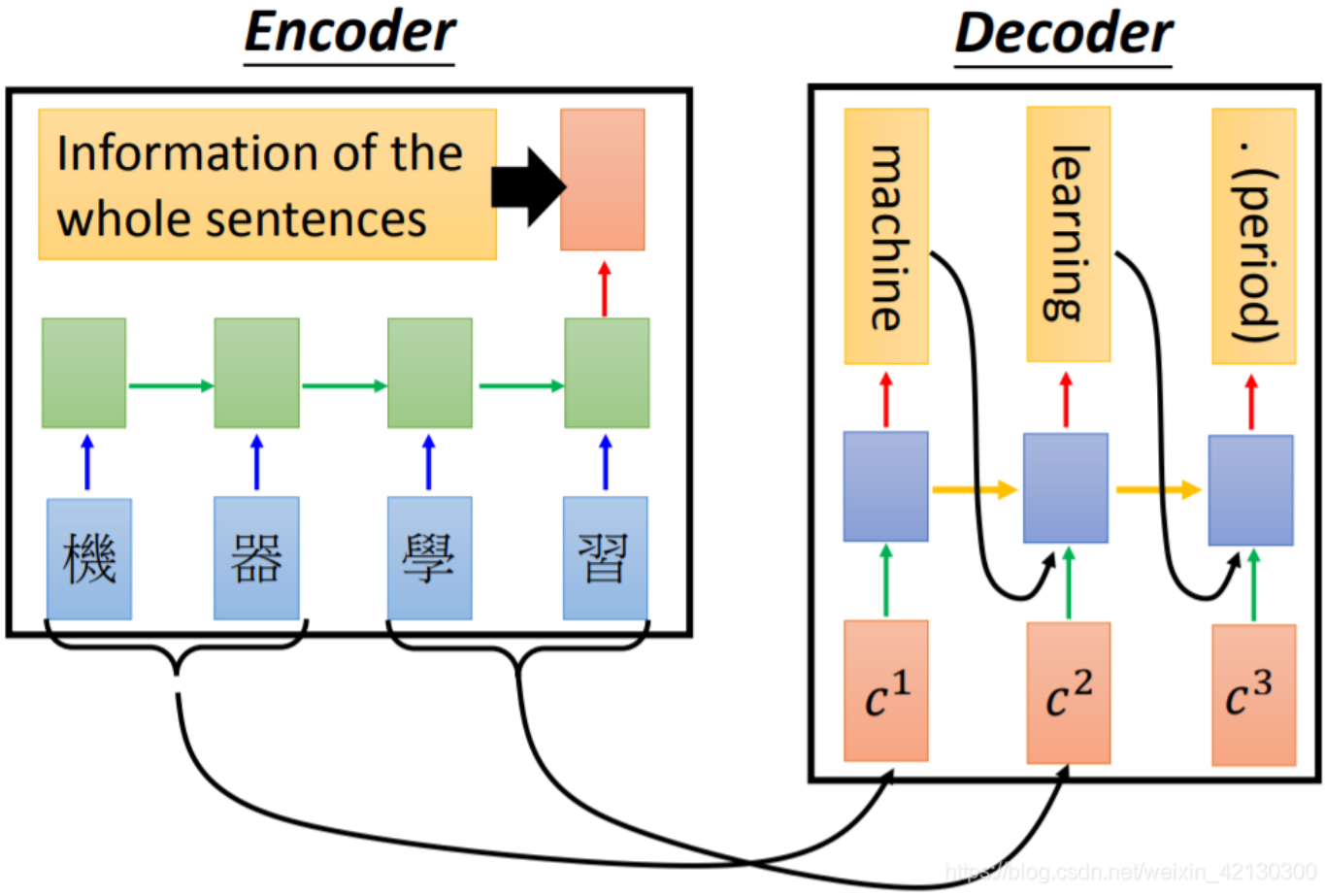


类似于RNN模型，输入的序列作为向量，训练后得到一个一个新的向量，可以代表它包含着输入序列的所有信息(encoder)，在输入的同时，进行输出。例如，在输入“机器”这个词后，这个模型就可以输出“machine”,在接受到“学习”后，机器可以输出“learning”。

- Represent the input condition as a vector, and consider the vector as the input of RNN generator
- E.g. Machine translation / Chat-bot



- Attention(Dynamic Conditional Generation)



- Tips for Generation

  如果要在Seq2seq上得到好的结果，需要加入合适的正则化项，比如：

$$\sum_i (\sigma - \sum_t \alpha_t^i)^2$$

三、Adversarial Attack(实验部分)

本节主要介绍如何使用Fast Gradient Sign Method (FGSM)来攻击训练好的模型。FGSM的算法流程如下：

$$x^{adv} = x + \varepsilon \cdot \mathtt{sign}(\nabla_x J(x, y_{true}))$$

where

x is the input (clean) image,

x^{adv} is the perturbed adversarial image,

J is the classification loss function,

y_{true} is true label for the input x.

假设我们拥有如下的数据集：一个包含200张224*224的RGB图片，labels.csv中保存的是每张图片正确的label.神经网络可能使用的模型有：VGG-16，VGG-19，ResNet-50，ResNet-101，DenseNet-121，DenseNet-169。本实验采用的是VGG-16模型。

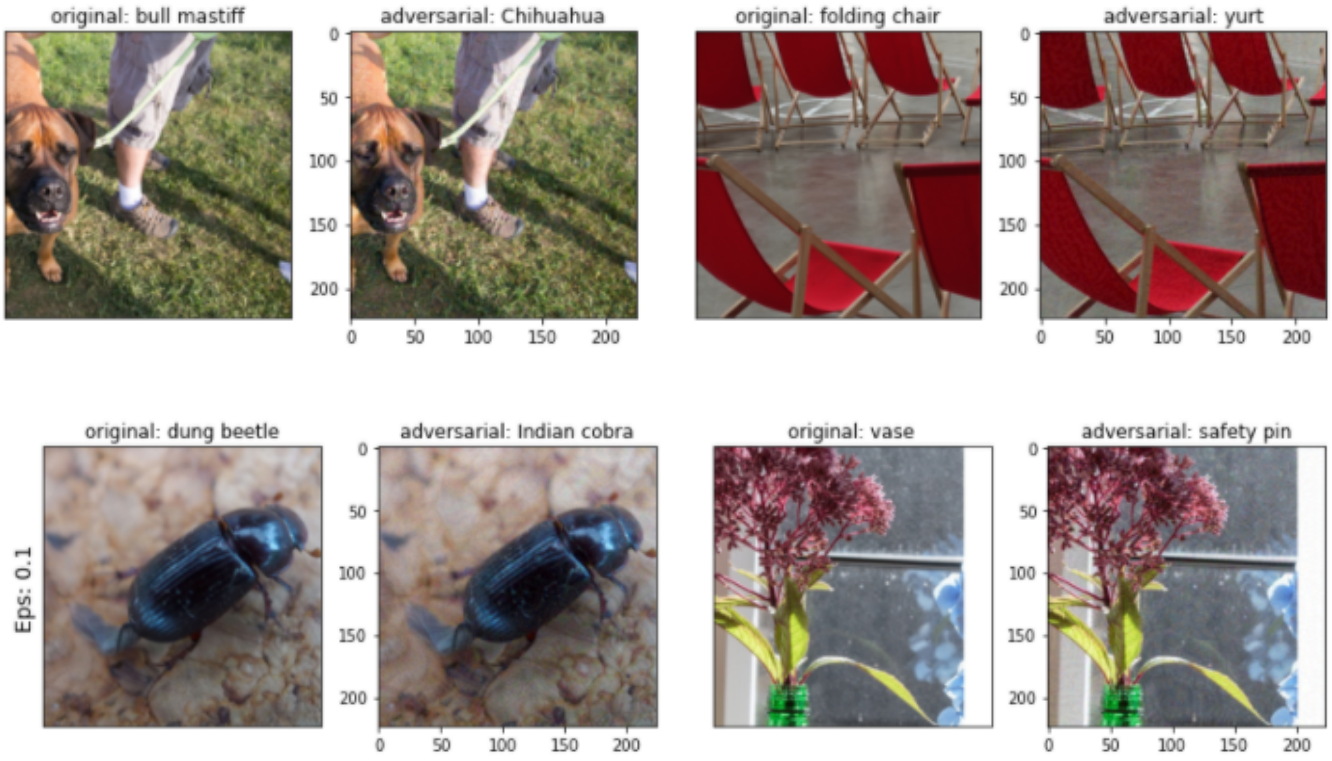
- 下載資料並解壓縮
  #下載資料
  !gdown --id '14CqX3OfY9aUbhGp4OpdSHLvq2321fUB7' --output data.zip
  #解壓縮
  !unzip -qq -u data.zip
  #確認目前的檔案
  !ls
- 导入需要的包
  import os
  # 讀取 label.csv
  import pandas as pd
  # 讀取圖片
  from PIL import Image
  import numpy as np
  import torch

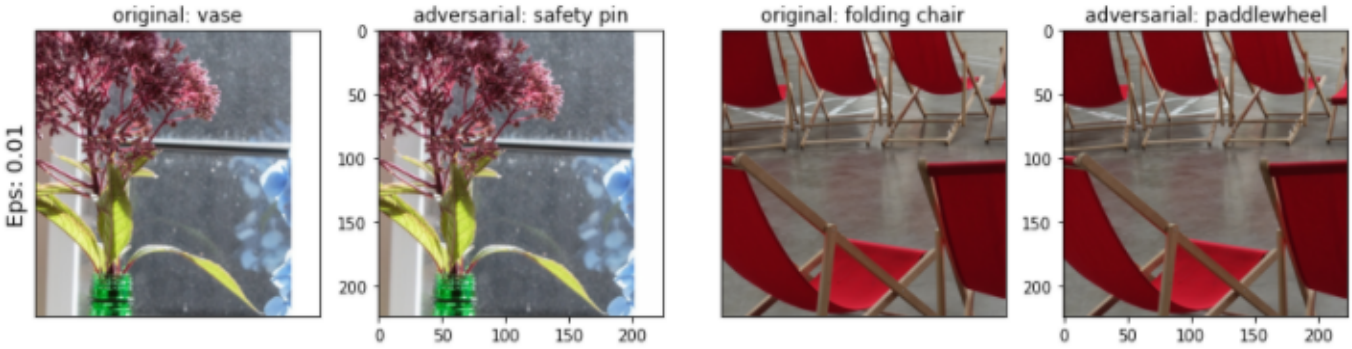
```
# Loss function
import torch.nn.functional as F
# 讀取資料
import torchvision.datasets as datasets
from torch.utils.data import Dataset, DataLoader
# 載入預訓練的模型
import torchvision.models as models
# 將資料轉換成符合預訓練模型的形式
import torchvision.transforms as transforms
# 顯示圖片
import matplotlib.pyplot as plt
device = torch.device("cuda")
```

- FGSM模型的实现如下：

```
# FGSM 攻擊
def fgsm_attack(self, image, epsilon, data_grad):
# 找出 gradient 的方向
sign_data_grad = data_grad.sign()
# 將圖片加上 gradient 方向乘上 epsilon 的 noise
perturbed_image = image + epsilon * sign_data_grad
return perturbed_image
```

经过攻击后，得到的实验结果如下：




Eps: 0.01
original: vase
adversarial: safety pin
original: folding chair
adversarial: paddlewheel
0
50
100
150
200