THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

CAM, Attention, and Transformers – COMP4423 Computer Vision

Xiaoyong Wei (魏驍勇)
x1wei@polyu.edu.hk

Department of Computing
電子計算學系

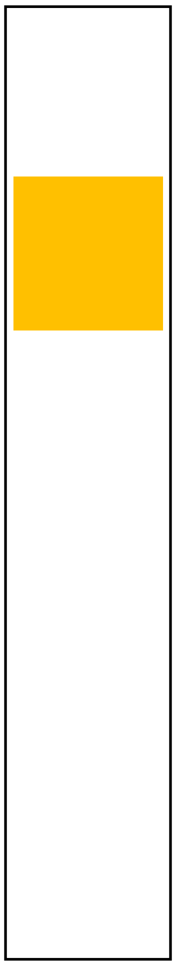
Opening Minds • Shaping the Future
啟迪思維 • 成就未來

Outline

- >From concepts to code
- >Datasets for deep learning
- >Class Activation Mapping (CAM)
- >Attentions
- >Self-Attentions, and Transformers

It’s time to link the concepts we learned to the actual code

Prepare the dataset by splitting it into batches



A Batch (subset)
BATCH_SIZE: #samples per batch
Shuffle: Shuffle the order of samples

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size=BATCH_SIZE,
                                          shuffle=True, num_workers=2)
```

Loading parallelization

Batch ID

Images of the batch

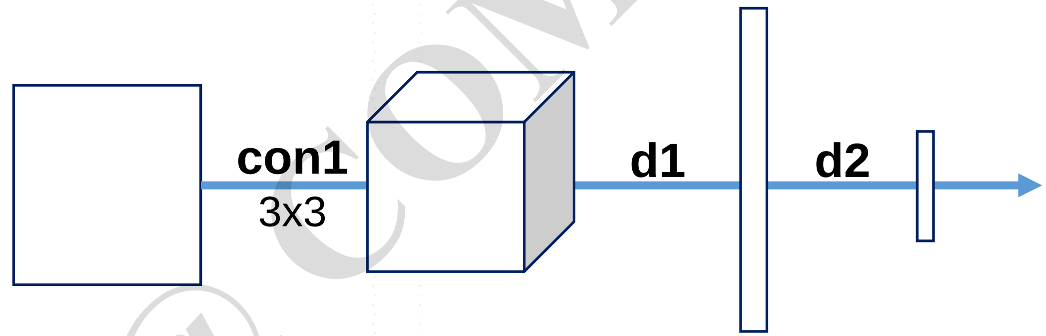
Labels of the images

```
for i, (images, labels) in enumerate(trainloader):
```

A Dataset

Design the networks

A Batch (subset)
BATCH_SIZE: 32
Image size: 28x28

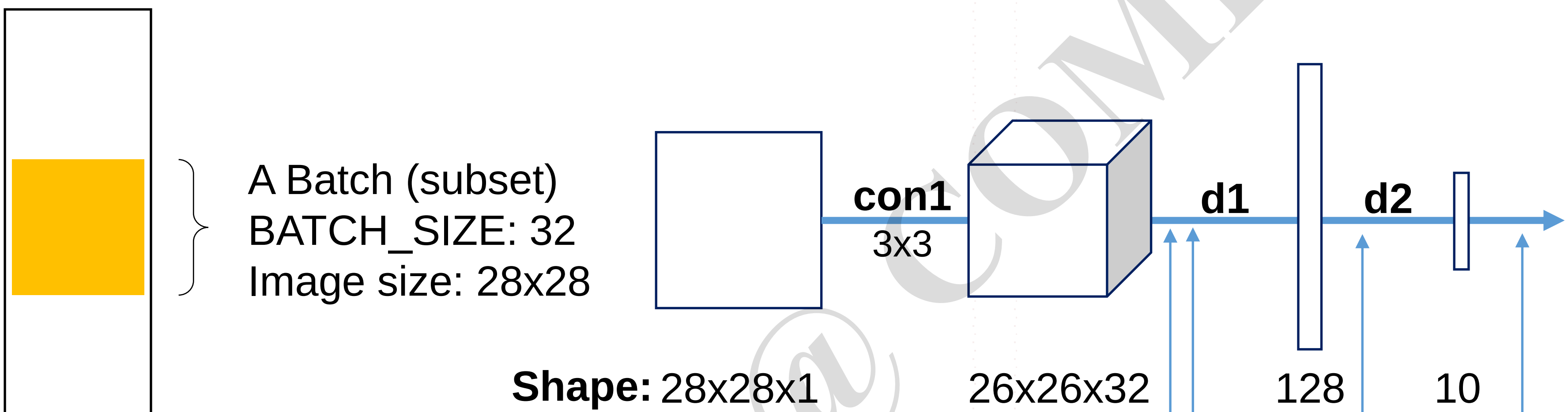


Shape: 28x28x1 26x26x32 128 10

```
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()

        # 28x28x1 => 26x26x32
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3)
        self.d1 = nn.Linear(26 * 26 * 32, 128)
        self.d2 = nn.Linear(128, 10)
```

A Dataset

Design the networks




```python
def forward(self, x):
    # 32x1x28x28 => 32x32x26x26
    x = self.conv1(x)
    x = F.relu(x)

    # flatten => 32 x (32*26*26)
    x = x.flatten(start_dim = 1)

    # 32 x (32*26*26) => 32x128
    x = self.d1(x)
    x = F.relu(x)

    # logits => 32x10
    logits = self.d2(x)
    out = F.softmax(logits, dim=1)
    return out
```

A Dataset

The Loss and Optimizer

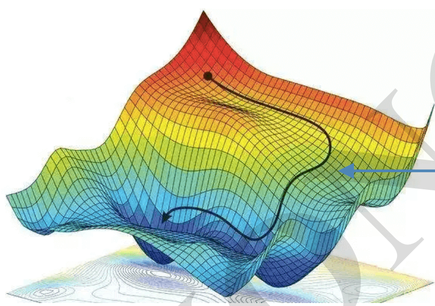
A Batch (subset)
BATCH_SIZE: 32
Image size: 28x28

con1
3x3

d1

d2

Shape: 28x28x1 26x26x32 128 10



$$H(P^*|P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTRIBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTRIBUTION}}$$

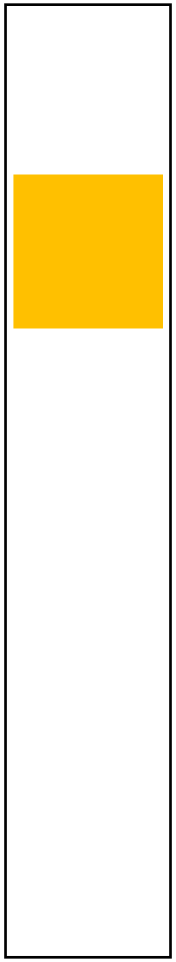
```
model = SimpleCNN()
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
```

$$a' = a' - \delta \frac{\partial L}{\partial a'}$$

A Dataset

Learning Rate

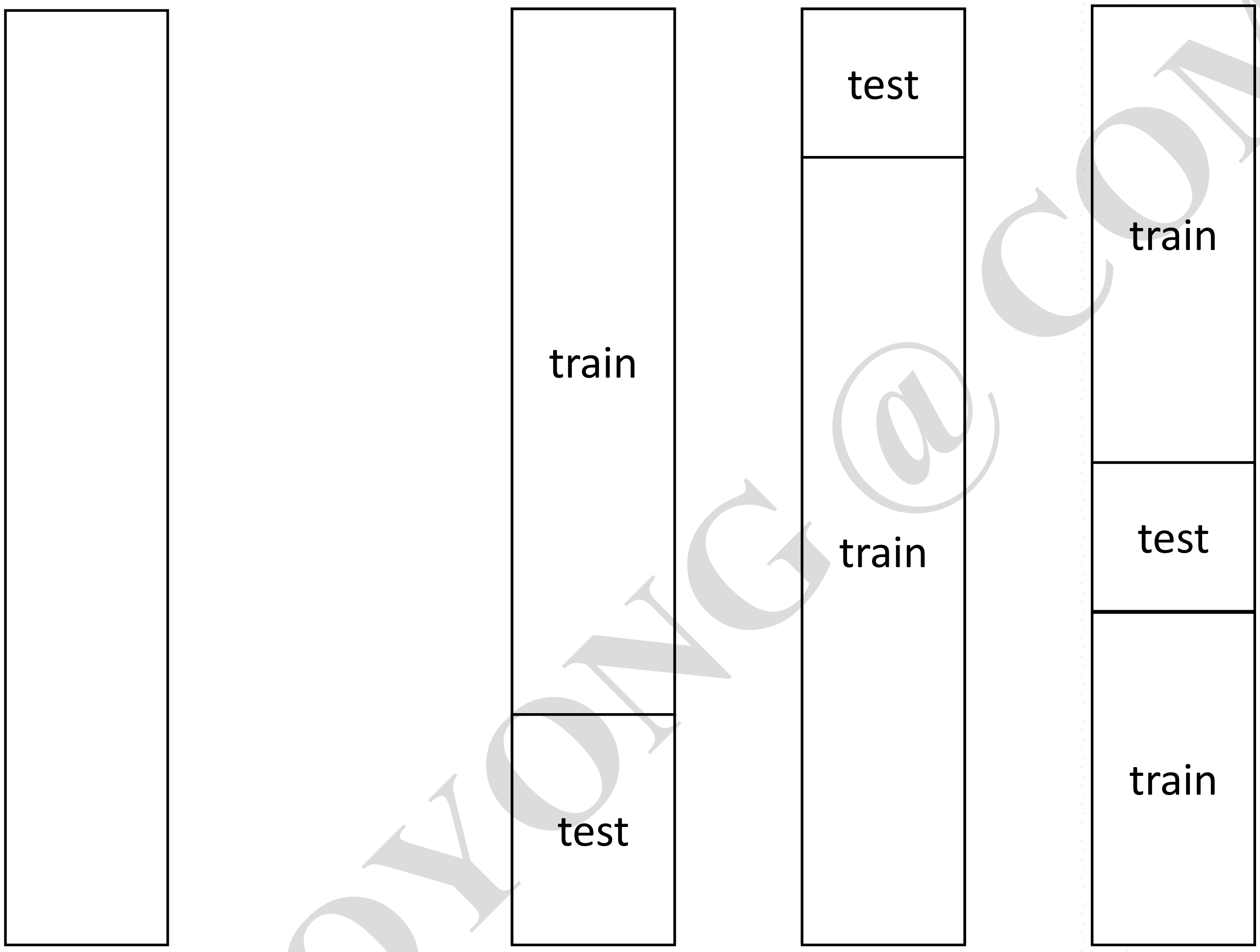
Get everybody into position



A Dataset

```
for epoch in range(num_epochs):
    train_running_loss = 0.0
    train_acc = 0.0

    model = model.train() # tell the model we're training

    ## training step
    for i, (images, labels) in enumerate(trainloader):

        images = images.to(device)
        labels = labels.to(device)

        ## forward + backprop + loss
        logits = model(images) # forward
        loss = criterion(logits, labels) # loss between the predicted and truth
        optimizer.zero_grad() # clean gradients
        loss.backward() # do backprobagation and udapte gradients

        ## update model params
        optimizer.step()

        train_running_loss += loss.detach().item()
        train_acc += get_accuracy(logits, labels, BATCH_SIZE)
```

N-Fold Cross Validation



A Dataset

```
scores = cross_val_score(model, feat_x, feat_y, cv=10)
```

The advantage of Deep Learning for most of us is that we can build complex models by simply stacking layers

To train the models, we need data, especially those labeled ones. However, data labeling is a costly process

The publicly available datasets are thus of great help

ImageNet

Images : 14197122

Classes : 21841

Tasks: Image classification

ImageNet is one of the largest and most classic CV public datasets. This dataset has driven the development of computer vision, but there are some problems in the dataset that the quality of the images is low, or the image might be wrong.

Source: ImageNet Large Scale Visual Recognition Challenge.

SOTA Performance on ImageNet

Backbone:Transformer

Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**CoAtNet-7**	90.88%		2440M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer JFT-3B
2	**ViT-G/14**	90.45%		1843M	✓	Scaling Vision Transformers			2021	Transformer JFT-3B
3	**CoAtNet-6**	90.45%		1470M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer JFT-3B
4	**ViT-MoE-15B** (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts			2021	Transformer JFT-3B

https://paperswithcode.com

CIFAR

CIFAR-100 :

Images:60000

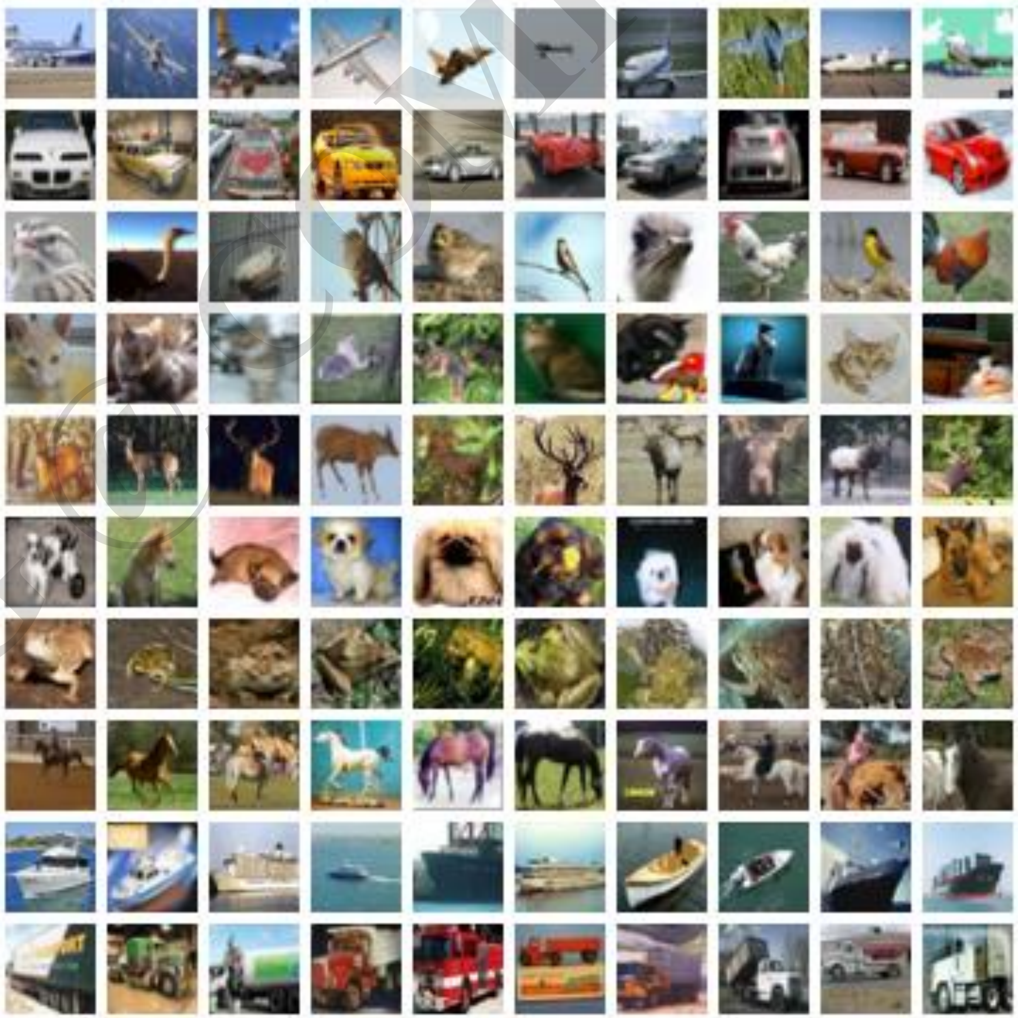
Classes:100

Tasks: Image classification

Sub dataset CIFAR-10:

Image:6000

Classes:10



The CIFAR dataset is more tidy, and its sub dataset CIFAR-10 is very suitable for getting started with image classification.

Source: Learning Multiple Layers of Features from Tiny Image.

SOTA Performance on CIFAR-100

Backbone: Transformer
EfficientNet

Rank	Model	Percentage correct	PARAMS	Extra Training Data	Paper	Code	Result	Year	Tags
1	**EffNet-L2** (SAM)	96.08		✓	Sharpness-Aware Minimization for Efficiently Improving Generalization			2020	EfficientNet
2	**ViT-H/14**	94.55±0.04		✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020	Transformer
3	**ViT-B-16** (ImageNet-21K-P pretrain)	94.2		✓	ImageNet-21K Pretraining for the Masses			2021	Transformer
4	**CvT-W24**	94.09		✓	CvT: Introducing Convolutions to Vision Transformers			2021	Transformer

https://paperswithcode.com

COCO

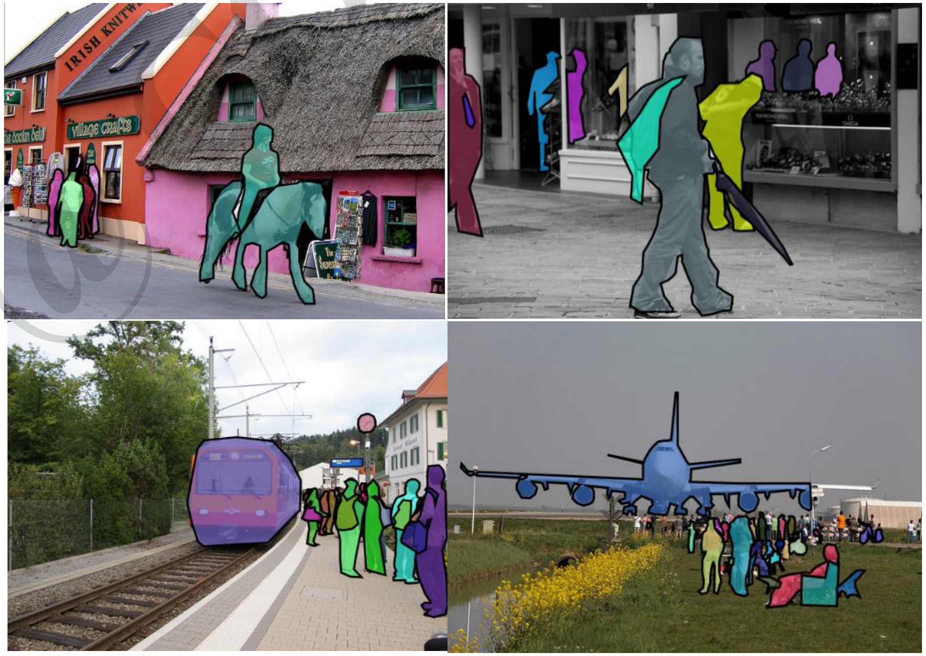
Images : 300K

Labels : 200K

Object : 80

Stuff : 91

Tasks: Image Detection / Image Segmentation



COCO is one of the largest image recognition datasets with fine annotations and a large number of images.

Source: ImageNet Large Scale Visual Recognition Challenge.

ADE20K

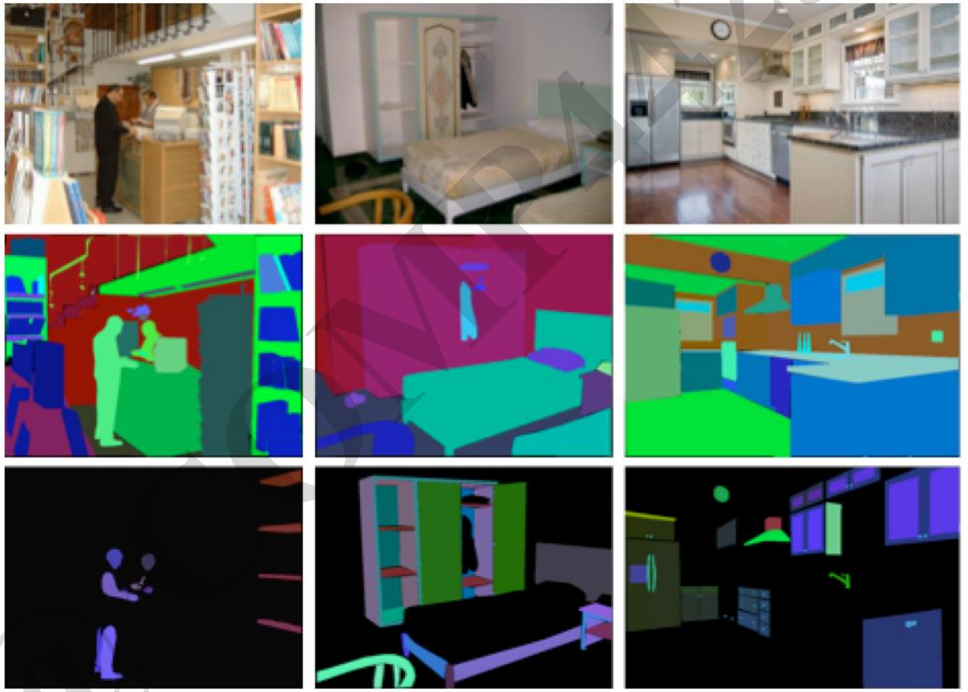
Images : 27574

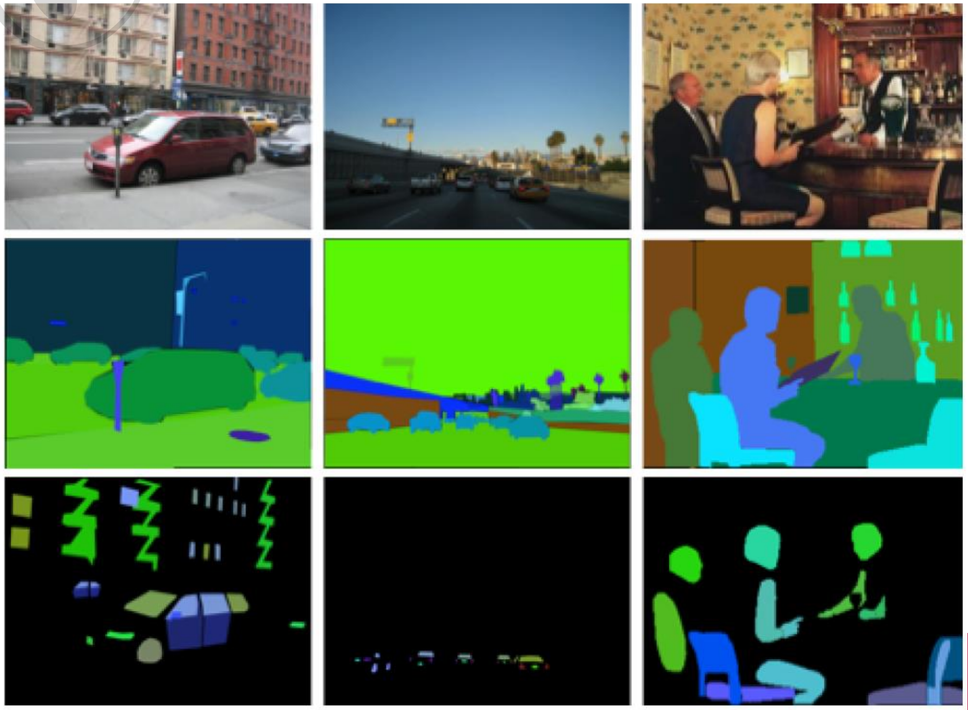
Classes : 21841

Object : 250

Tasks: Image Segmentation

Ade20k data set is one of the most commonly used data sets for image segmentation, which contains rich indoor and outdoor scenes and objects.





Department of Computing
電子計算學系

Source: ImageNet Large Scale Visual Recognition Challenge.

Great! We have models and datasets. But how can we tell if the models are working properly?

Beside the Accuracy, Precision, and Recall which we used to evaluate the performance overall, we cannot tell exactly how it works

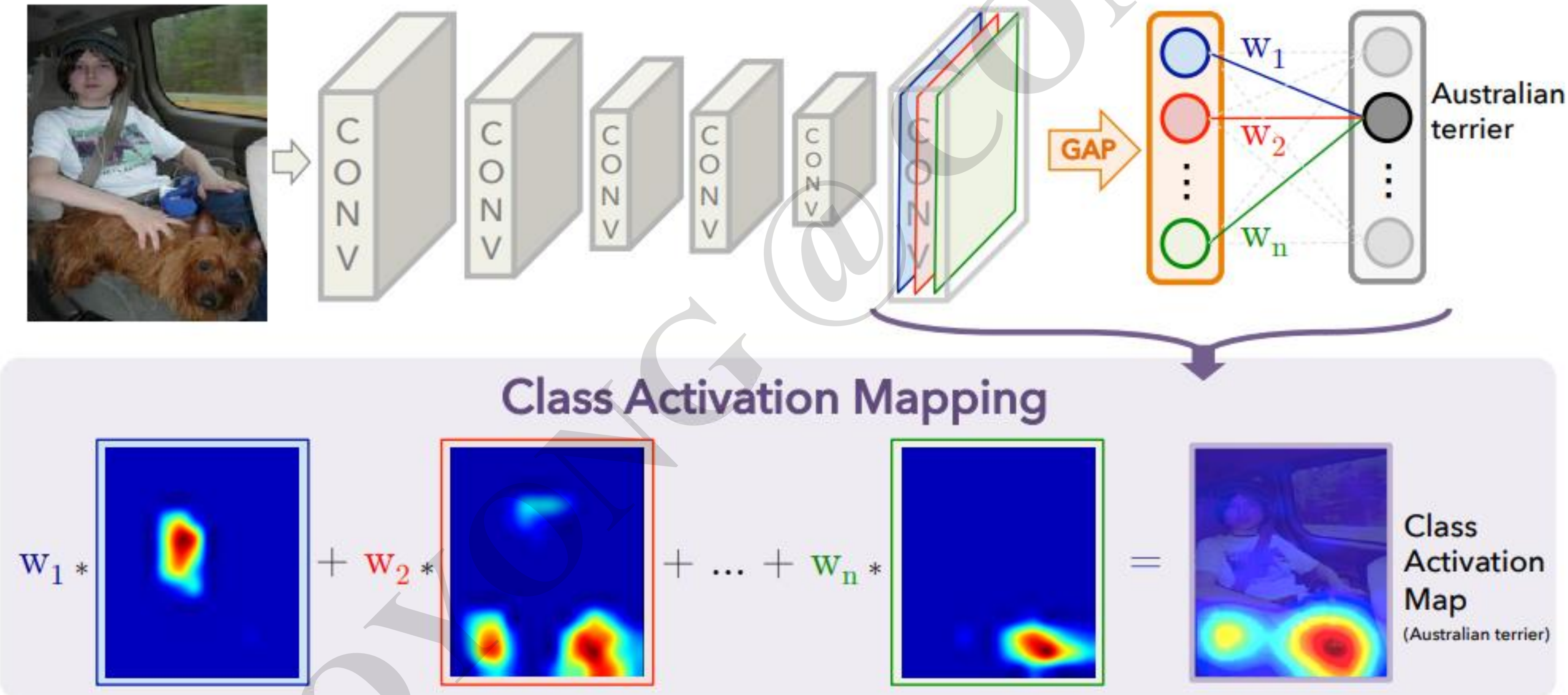
That's the reason deep models are considered as a "**Blackbox**"

Luckily, there are some “probes” that we can use

Class Activation Mapping (CAM)

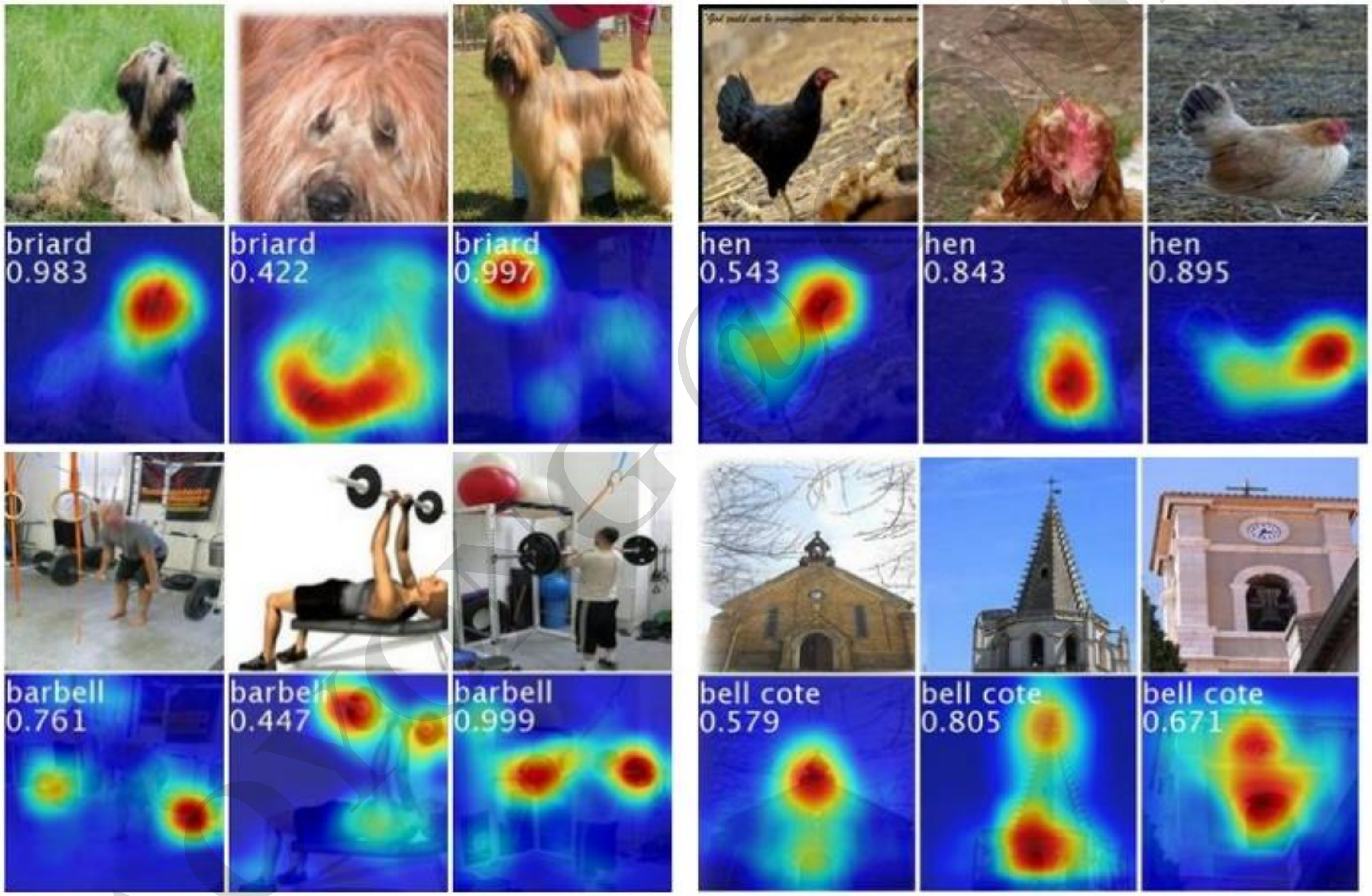
Source: Learning Deep Features for Discriminative Localization

Class Activation Mapping (CAM)



Source: Learning Deep Features for Discriminative Localization

Class Activation Mapping (CAM)



Source: Learning Deep Features for Discriminative Localization

Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM), uses the gradients of any target concept (say ‘dog’ in a classification network or a sequence of words in captioning network) flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

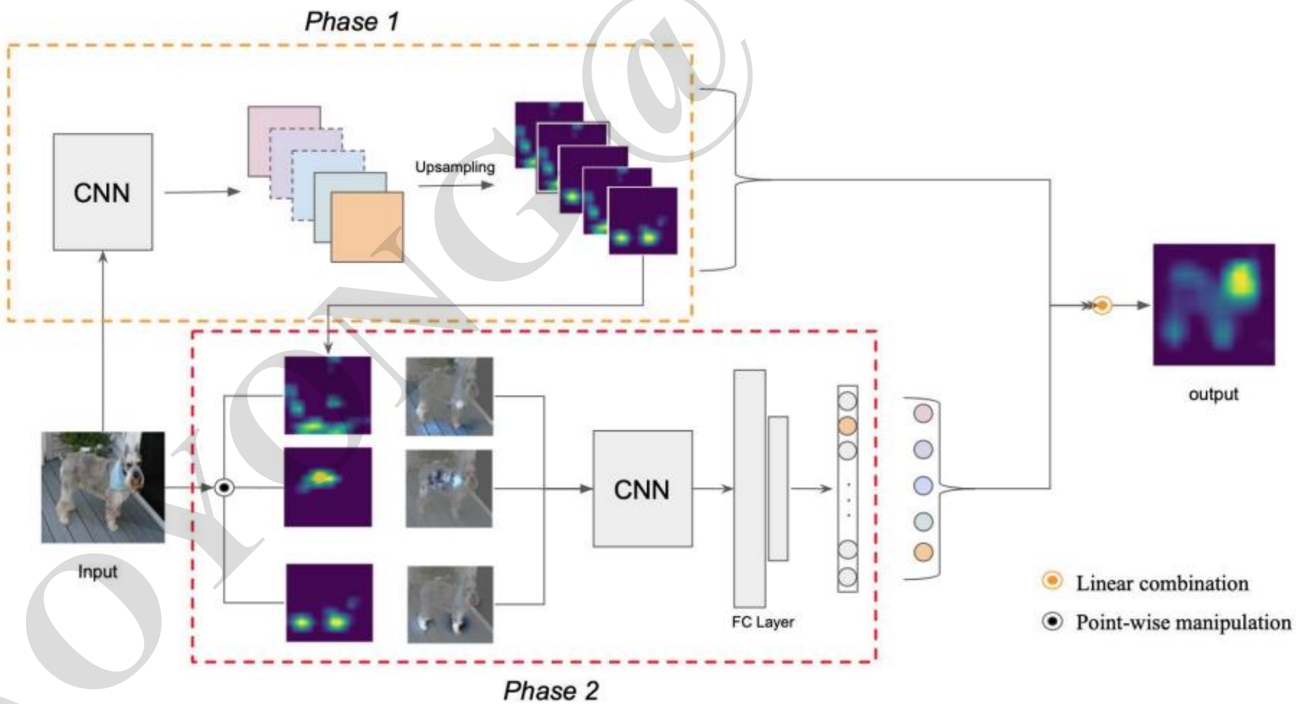
Unlike CAM, Grad-CAM is applicable to a wide variety of CNN model-families without architectural changes or re-training.

(a) Original Image (b) Guided Backprop ‘Cat’ (c) Grad-CAM ‘Cat’

Source: Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Score-CAM

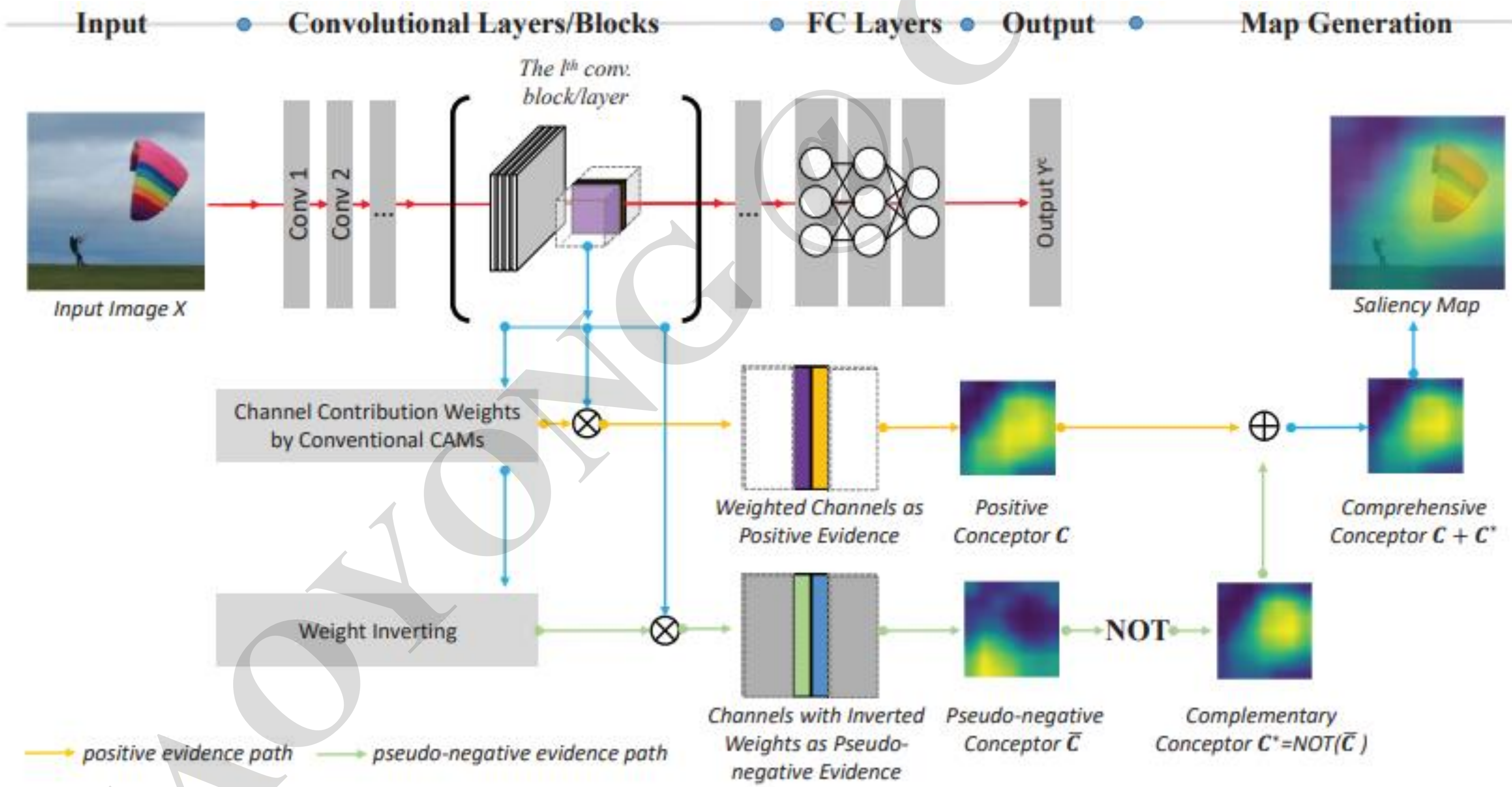
Unlike previous class activation mapping based approaches, Score-CAM gets rid of the dependence on gradients by obtaining the weight of each activation map through its forward passing score on target class, the final result is obtained by a linear combination of weights and activation maps. Score-CAM achieves better visual performance and fairness for interpreting the decision making process.



Source: Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks

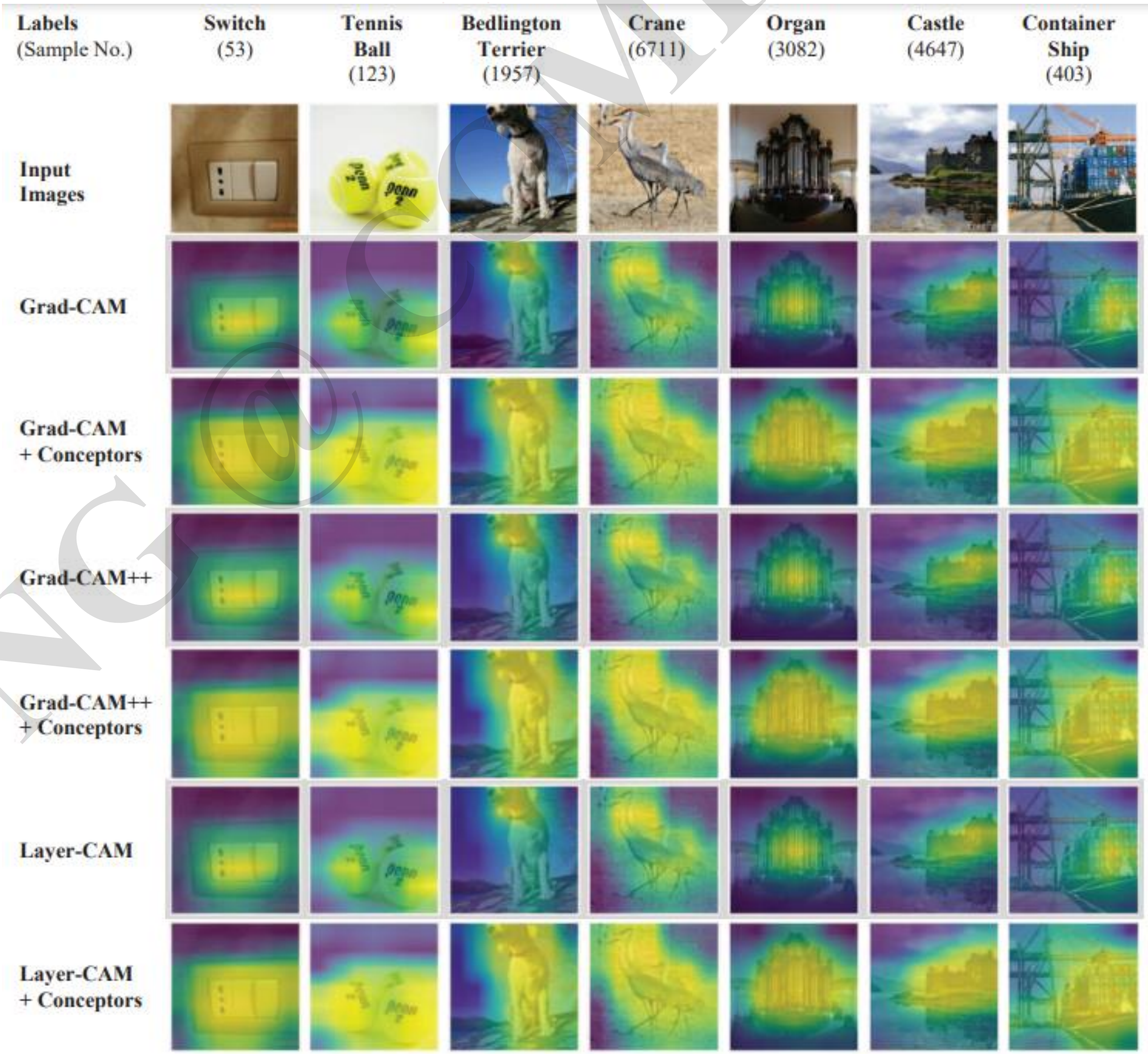
Conceptor-CAM

In Conceptor-CAM, we unify the previous CAMs into a framework, in which they are different from each other in the way of modeling the channel relations. We find the inner-channel relations are ignored in those CAMs. So we propose Conceptor-CAM to model both inter- and inner-relation in one shot.



Source: Conceptor Learning for Class Activation Mapping

Conceptor-CAM



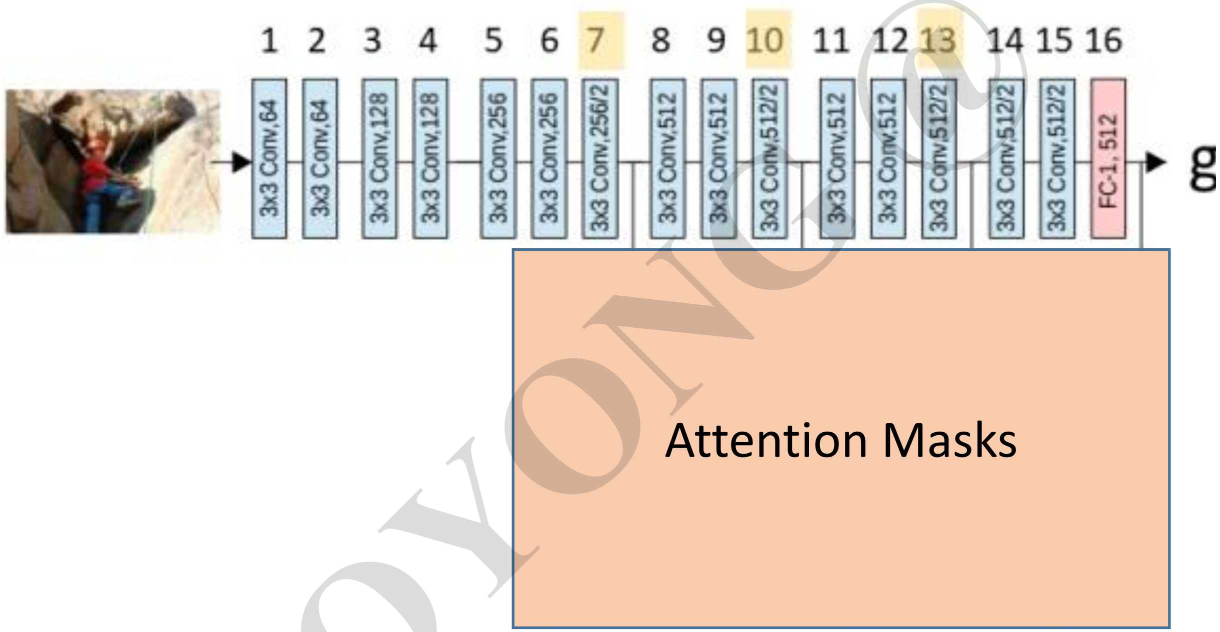
Conceptor-CAM is in fact compatible to other CAMs. We can use Conceptors to “regulate” the relations in those CAMs for better performance

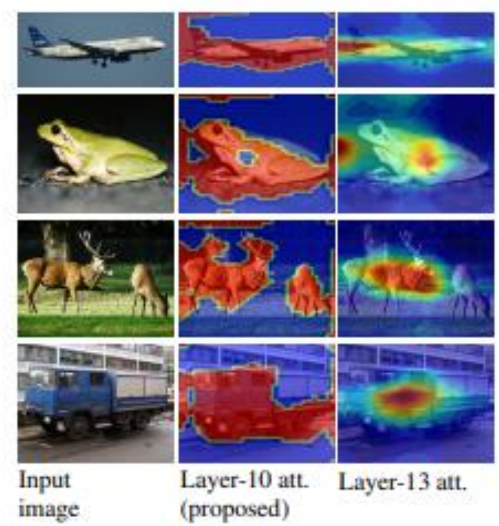
Source: Conceptor Learning for Class Activation Mapping

While we can "see" how well the objects are focused or not, can we "help" models to **focus**?

Attention

When we think about the English word “Attention”, we know that it means directing your focus at something and taking greater notice. The Attention mechanism in Deep Learning is based on this idea, and it pays greater attention to certain factors when processing the data.

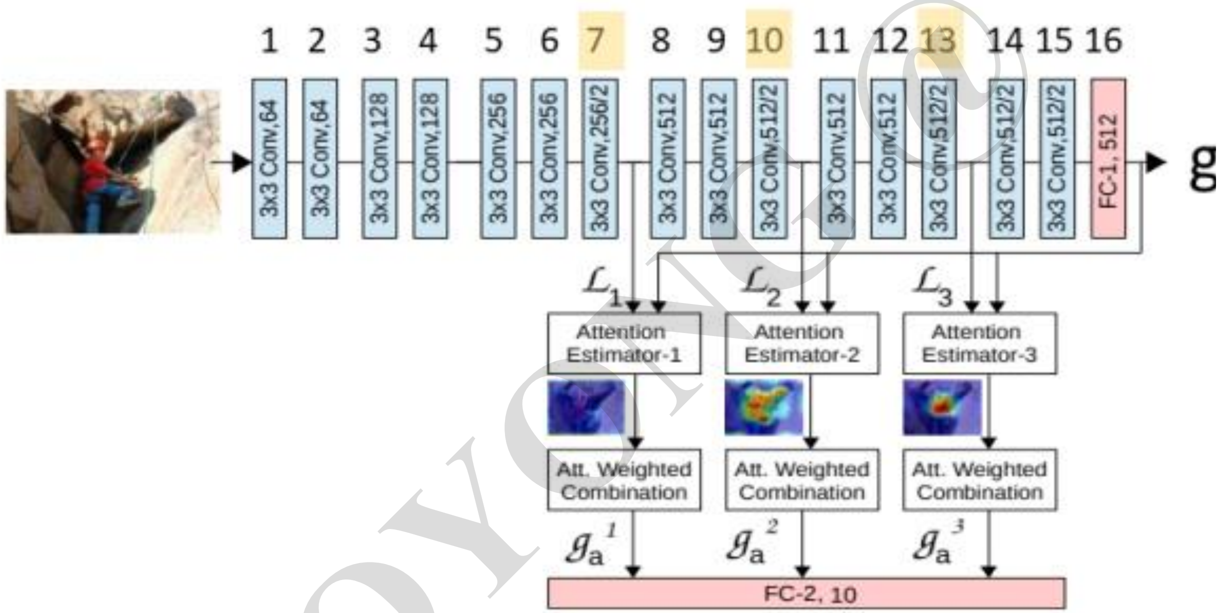


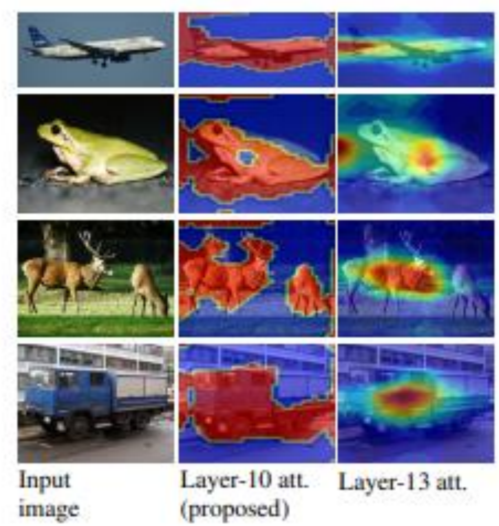


Source: Learn to pay attention

Attention

When we think about the English word "Attention", we know that it means directing your focus at something and taking greater notice. The Attention mechanism in Deep Learning is based on this idea, and it pays greater attention to certain factors when processing the data.

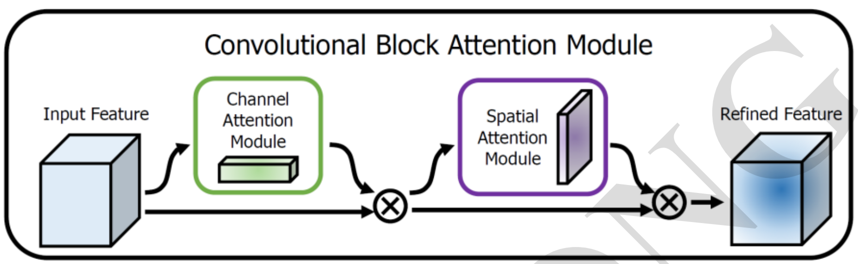




Source: Learn to pay attention

CBAM: Convolutional Block Attention Module

CBAM learns what and where to emphasize or suppress and refines intermediate features effectively.



The overview of CBAM

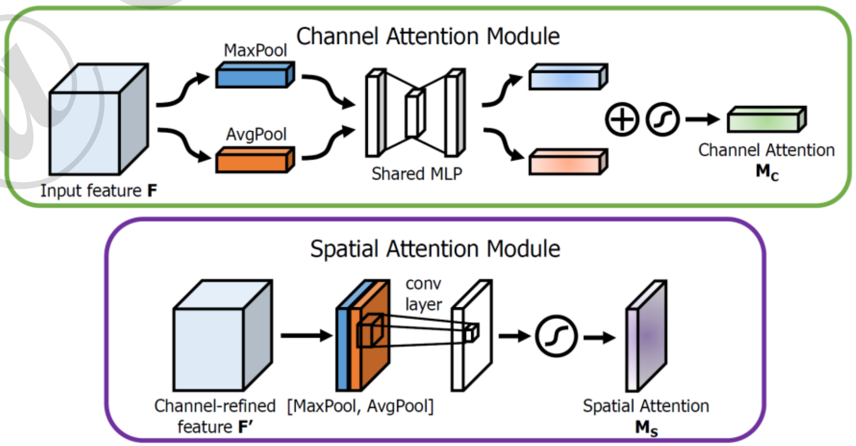


Diagram of each attention sub-module

Woo S, Park J, Lee J Y, et al. Cbam: Convolutional block attention module, Proceedings of the European conference on computer vision (ECCV). 2018: 3-19.

Let's go one step further

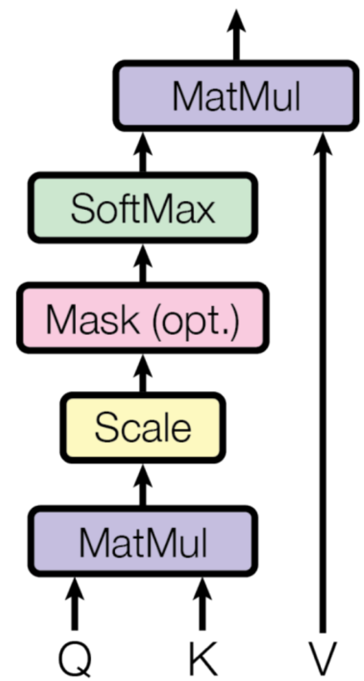
Self Attention

Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need, 2017.

Self Attention

- > Self-Attention calculates a weight according to the input data to represent the attention of each position in the data sequence to other different positions. The figure on the right is a simple example.

- > The specific implementation of self attention is to map three matrices Q, K and V using the input data. The dot products of Q and K are normalized by softmax to represent the weight matrix and V represents the value matrix. The formula is as follows:



$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

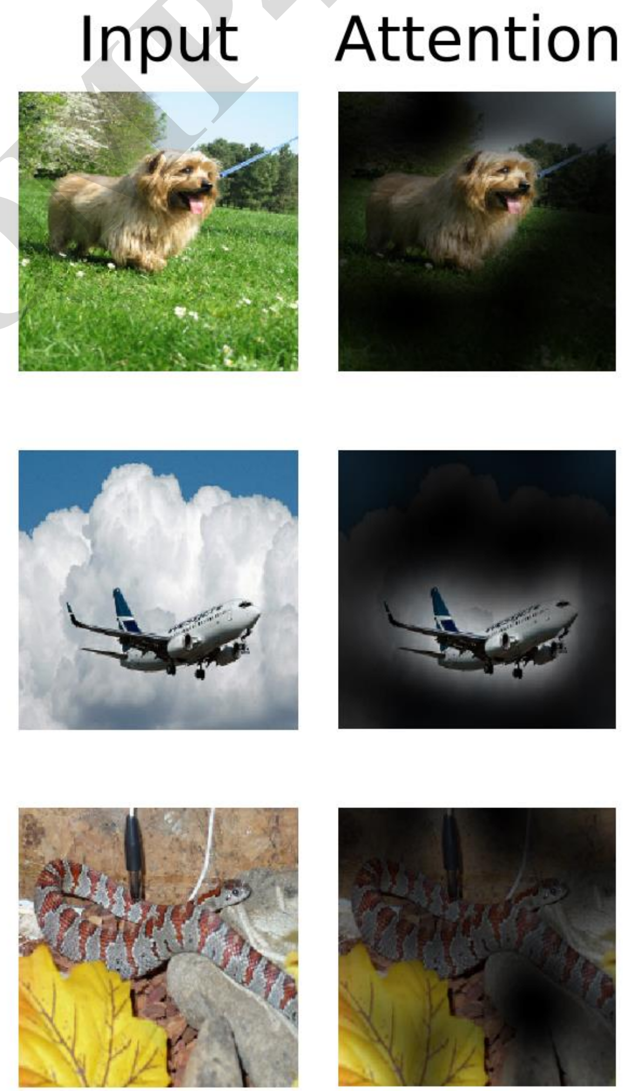
Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need, 2017.

Self Attention

> Self-Attention calculates a weight according to the input data to represent the attention of each position in the data sequence to other different positions. The figure on the right is a simple example.

> The specific implementation of self attention is to map three matrices Q, K and V using the input data. The dot products of Q and K are normalized by softmax to represent the weight matrix and V represents the value matrix. The formula is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



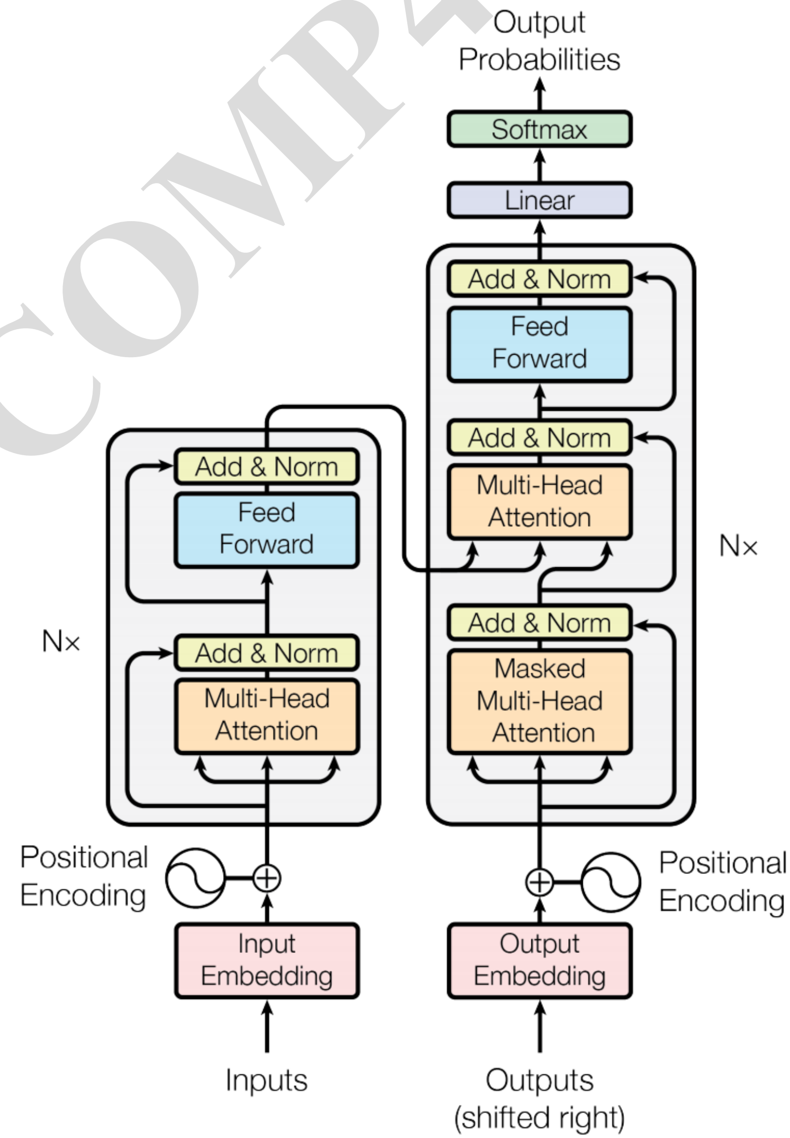
Source: AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need, 2017.

Transformer

Transformer is an Encoder-Decoder architecture based on Self-Attention.

Transformer was first used in the field of natural language processing with success, and then used in computer vision tasks.

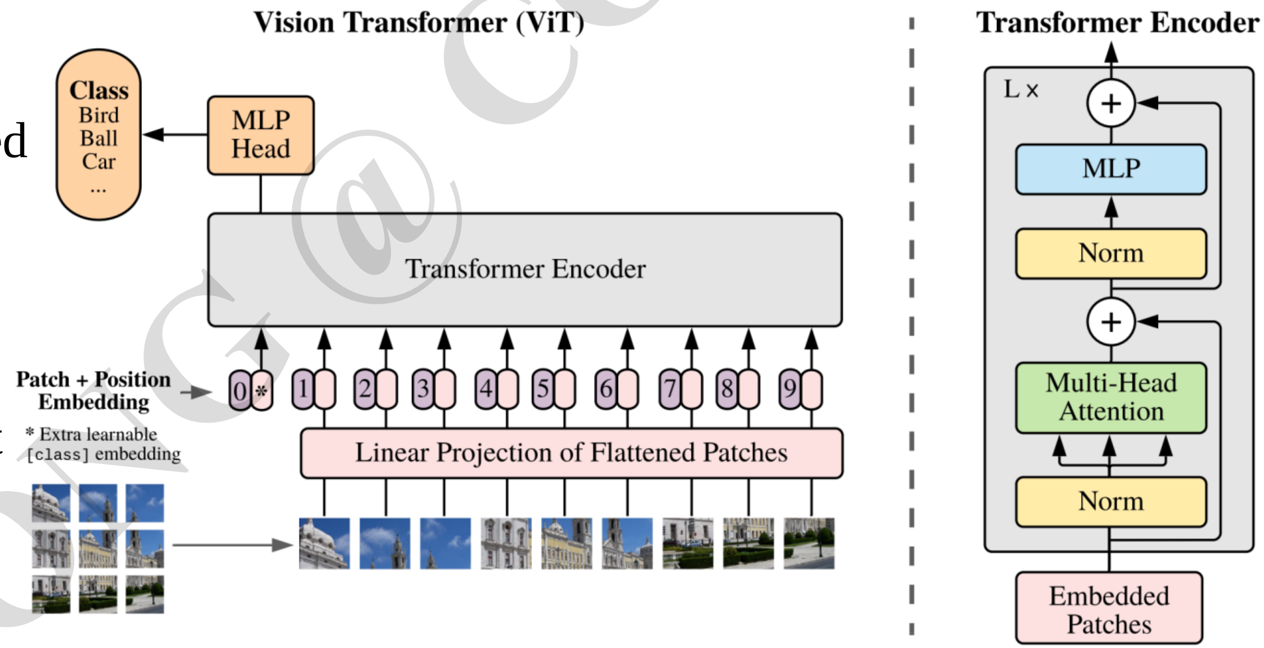


Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need, 2017.

Vision Transformer (ViT)

> The image will be segmented into patches before entering the model, and the patches are reshaped as 2D vectors.

> The model is a standard transformer Encoder architecture, and the output of Encoder gets the probability of each classification after MLP layer.



Dosovitskiy A , Beyer L , Kolesnikov A , et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020.

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學
Department of Computing
電子計算學系

Thank you!