

Alexnet

这个CNN模型在ImageNet数据集上取得了突破性的成绩

- 首次使用ReLU激活函数
- 首次使用Dropout技术防止过拟合
- 首次使用数据增强技术

GoogleNet

这个CNN模型在ImageNet数据集上取得了优异的成绩

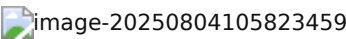
- 首次使用Inception v1模块
- 首次使用辅助分类头
- 首次使用全局平均池化

GoogLeNet模型在ImageNet数据集上取得了优异的成绩，其核心模块是Inception v1

Inception v1



- Inception v1模块由四个并行分支组成：1x1卷积、3x3卷积、5x5卷积和池化操作
- 每个分支的输出经过ReLU激活函数并归一化后，再按通道数相加



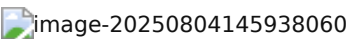
这个模块在ImageNet数据集上取得了优异的成绩



- GoogLeNet v3模型在ImageNet数据集上取得了优异的成绩，其核心模块是Inception v3
- Inception v3模块由两个并行分支组成：一个分支包含深度可分离卷积和最大池化，另一个分支包含深度可分离卷积和平均池化

Resnet

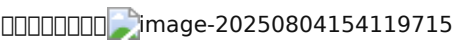
2015年，微软研究院的何恺明等人提出了ResNet模型，该模型在ImageNet数据集上取得了优异的成绩。ResNet模型的核心思想是引入残差块（Residual Block），使得网络可以训练得更深。ResNet模型的数学表达式为：
$$H(x) = x + F(x)$$
其中， x 表示输入特征， $F(x)$ 表示残差函数。ResNet模型在ImageNet数据集上取得了优异的成绩，其核心模块是ResNet Block



ResNet Block

BasicBlock 是最简单的ResNet Block，由两个卷积层组成

Bottleneck 是更复杂的ResNet Block，由三个卷积层组成，中间层使用1x1卷积进行降维



• BasicBlock

- **Layer2** 即 **layer** 即 **Block** 即 **Block** 即 **Block**
- 3×3 卷积核 64
- **block**
- 即
 1. **1** **layer**
 2. $2 \times \text{layer} \rightarrow 1 \times \text{block} \rightarrow \text{block} \rightarrow \text{layer}$

• BottleNeck

- **Layer** 即 **Block** 即 **Block** 即 **Block**
- 3×3 卷积核 $1 \times 1 \times 3 \times 1 \times 1$
- 1×1 卷积核
- 3×3 卷积核
- 1×1 卷积核 4
- 即
 1. $\text{layer} \rightarrow 1 \times \text{block} \rightarrow \text{block} \rightarrow \text{layer}$
 2. 1×1 卷积核 4
 3. 1×1 卷积核

-
- **layer** **block** **stride=2** **block** **stride=1**

ResNet



image-20250805173724557

VGG

3×3 卷积核

3×3 卷积核 5×5 , 7×7

- 3×3 卷积核
- $1 \times \text{stride} = 1$ **padding=1**
- 3×3 卷积核



image-20250805182841345

MobileNet

****95%****
\$1\times 1\$

[illegible]

- 444

5/5

□□□□

ReLU6 $\text{ReLU6}(x) = \min(\text{ReLU}(x), 6)$

ResNet

- 1111

□□□□□ReLU□□□□□□□□□□□□□□□□□□□□ReLU□□□□□□

111

- [illegible]

- MobileNetV1: 0 ReLU, 0, 0 ReLU

Vit Vision Transformer

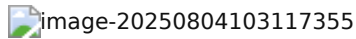
Transformer

5/5

1. `tokens = tokenizer.tokenize('The patches are on the wall')`
2. `tokens = tokenizer.tokenize('The patches are on the wall')[:768]` (768 tokens max)
3. `tokens = tokenizer.tokenize('The patches are on the wall')[:768]` (768 tokens max)
4. `tokens = tokenizer.tokenize('The patches are on the wall')[:768]` (768 tokens max)

111

1.
2. CNNInstructive Bias
3. patch






Deit

Vit

□□□□distillation□

- CNN-Vit
- DeiT-Vit encoder [distill] Token
 - [distill]
 - [CLS]
- hard distillation
- soft distillation KL

□□□□□□

- 
- 
- 

□□□□

Semantic Segmentation

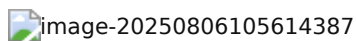
-
-

Instance Segmentation

- [illegible]

□□□□ Panoptic Segmentation□

- [illegible]



FPN Feature Pyramid Network

tmkaiming

問題を解決する

- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す


問題の定義とデータの収集

問題の定義

- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す

データの収集

- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- $\$1 \times 1$ の畳み込み

 image-20250809155500175

FCN (Fully Convolutional Network)

問題の定義

データの収集


- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す

問題の定義とデータの収集

- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
 - $\$1 \times 1$ の畳み込みと softmax 関数
- 問題の定義
- 問題の定義

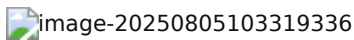
FCN の構築

- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
 - FCN-32s: 32 個の畳み込み層
 - FCN-16s: 2 個の畳み込み層 (pool4) と 16 個の畳み込み層
 - FCN-8s: 2 個の畳み込み層 (pool4) と 2 個の畳み込み層 (pool3) と 8 個の畳み込み層

 image-20250809162635744

U-Net

- U-Net: 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義、データの収集、モデルの構築、評価、改善のサイクルを繰り返す
- 問題の定義



1111

- [illegible]

111

Unet Unet

GoogleNet

[illegible]

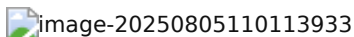
- □□□□□□□□□□□□□□□□
- □□□□□□
- □□□□□□□□□□□□□□□□
- □□□□□□

IoU□□□□□

$\{A\cup B\}$
 $\$ \$$
 $\text{IoU} = \frac{A\cap B}{A\cup B}$

IoU

- [illegible]



Dice Loss

Dice $\text{Dice Loss} = 1 - \frac{2 \cdot |\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A}| + |\mathcal{B}|}$

- $\$A\$$ 0~1
- $\$B$ 01
- $\$|A \cap B|$
- $\$|A| + |B|$

[illegible]

□□□□□□□□□□□□□□□□

BCE Loss

$$\text{DiceLoss} = - \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

- y_i
- p_i
- N

1. 1000 classes
 2. 1000 classes

Deeplab

1000 classes

v2 **v3** **backbone** **ResNet**

1000

1. 1000 classes
2. 1000 classes

LargeFov

1000

Deeplab 1000 classes

- 1000 classes
- 1000 classes

VGG

1. VGG backbone FC6 7×7
2. 3×3 $r=12$ 7×7
3. VGG backbone 1×1

ASPP model V3

1000 classes

1000 classes

 image-20250807104036319

backbone

1000

- 1×1
- $12 \times 3 \times 3$
- $24 \times 3 \times 3$
- $36 \times 3 \times 3$
- 1000

channel 1×1

 image-20250807104102110

- 16 $\times 3 = 48$
- 8 $\times 3 = 24$

V2 `poly = lr \times (1 - \frac{iter}{max_iter})^{\{power\}}`

- `$power$` 0.9-2.0
- `$iter$`
- `max_iter`

1. 1k~2k
- 2.
3. SVM
- 4.

1. Support Vector Machine (SVM)
2. Input/Output (IO)
3. Input/Output (IO)

[illegible]


image-20250807113641978

 image-20250808100134981

$$\hat{G}_x = P_{wd_x}(P) + P_x \setminus \hat{G}_y = P_{hd_y}(P) + P_y \setminus \hat{G}_w = P_{wexp}(d_x(P)) \setminus \hat{G}_h = P_{hexp}(d_x(P))$$

- P_x, P_y, P_w, P_h 输入特征图 x, y
- $\hat{G}_x, \hat{G}_y, \hat{G}_w, \hat{G}_h$ 输出特征图 x, y

Multi-task loss

$$L(p, \mu, t^{\mu}, v) = \underbrace{L_{cls}(p, \mu)}_{\text{classification}} + \underbrace{\lambda L_{loc}(t^{\mu}, v)}_{\text{location}} + \underbrace{\lambda L_{reg}(v)}_{\text{regression}}$$

- p 输入特征图 $p = (p_0, \dots, p_k)$
- μ 输入特征图 μ
- t^{μ} 输入特征图 t^{μ} 输入特征图 t^{μ}
- v 输入特征图 $v = (v_x, v_y, v_w, v_h)$
- λ 输入特征图 λ
- L_{cls} 输入特征图 L_{cls}
- L_{loc} 输入特征图 L_{loc}
- L_{reg} 输入特征图 L_{reg}

$$L_{cls}(p, \mu) = -\log\{p_{\mu}\}$$

$$L_{loc}(t^{\mu}, v) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L_1}(t^{\mu}_i - v_i)$$


$$x = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

$$v_x = \frac{G_x - P_x}{P_w} \setminus v_y = \frac{G_y - P_y}{P_h} \setminus v_w = \ln\{\frac{G_w}{P_w}\} \setminus v_h = \ln\{\frac{G_h}{P_h}\}$$

- G_x, G_y, G_w, G_h 输入特征图 x, y
- 输入特征图 x, y

ROI pooling

1. 输入特征图 x
2. **ROI** 输入特征图 x ROI 输入特征图 x
3. 输入特征图 x ROI 输入特征图 x
4. 输入特征图 **max pooling**

 image-20250811180354848

Faster R-CNN

tm kaiming

RPN + Fast R-CNN

1. 输入特征图 **1k~2k**
2. 输入特征图 **RPN**
3. 输入特征图 **ROI pooling** 7×7

RPN Region Proposal Network

输入特征图 x

1. **RPN Backbone**
2. 输入特征图 x

- [illegible]

3. RPN\$1\times 1\$\input type="text"/>

- 2k**□□□□□□sigmoid□□softmax

- 4k

RPN anchor 250 1:1

RPN **anchor** **2w** **2k** **Fast-RCNN**

$$\begin{aligned} \mathbb{E} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \underbrace{\frac{1}{N_{\text{cls}}} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \mathbb{E}_{\mathbf{p}_i, \mathbf{p}_i^*} \{ \text{loss}(\mathbf{p}_i, \mathbf{p}_i^*) \}}_{\text{loss}} + \\ \underbrace{\frac{1}{\lambda} \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \mathbb{E}_{\mathbf{p}_i, \mathbf{p}_i^*} \{ \text{loss}(\mathbf{p}_i, \mathbf{p}_i^*) \}}_{\text{loss}} \end{aligned}$$

- `p_i` `anchor`
- `p_i^{*}` `10`
- `t_i` `anchor`
- `t_i^{*}` `anchor`
- `N_{cls}` `mini-batch`
- `N_{reg}` `anchor`
- `L_{reg}` `$smooth$`
- `L_{cls}`

☐ Faster R-CNN ☒ mask ☐ ROI Pooling ☐ ROI Align

$$\text{BCE-Loss}_{L_{\text{mask}}} = L_{\text{RPN}} + L_{\text{Faster_rcnn}} + L_{\text{mask}}$$

ROI Align

ROI Pooling

ROI Pooling

- ROI pooling 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
 - 它接收一个特征图和一个 ROI 坐标。
 - 它返回 ROI 区域内的特征值。

ROI Align

- ROI pooling 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
- ROI pooling 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
- ROI pooling 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
- ROI pooling 是用于从特征图中提取感兴趣区域 (ROI) 的操作。

Mask

FPN Mask



image-20250811173140965

Mask

- Mask 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
 - 它接收一个特征图和一个 ROI 坐标。
 - 它返回 ROI 区域内的特征值。



image-20250811175302080

- Mask 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
 - 它接收一个特征图和一个 ROI 坐标。
 - 它返回 ROI 区域内的特征值。



image-20250811175327478

Mask R-CNN

Mask R-CNN

- Mask R-CNN 是用于从特征图中提取感兴趣区域 (ROI) 的操作。
 - 它接收一个特征图和一个 ROI 坐标。
 - 它返回 ROI 区域内的特征值。

YoloV5

Backbone DarkNet

YoloV7

YoloV8

Clip

Blip

数据集

- 数据集
- 数据集

COCO数据集

COCO数据集使用 `pycocotools.coco` 与 COCO 数据集的 `annotations` 文件

数据集

方法	描述	代码
<code>.getImgIds()</code>	获取所有ID	<code>train_coco.getImgIds()</code>
<code>.loadImgs(ids)</code>	加载ID对应的数据	<code>train_coco.loadImgs([1,2])</code>
<code>.getAnnIds(imgIds=[])</code>	获取所有ID	<code>train_coco.getAnnIds(imgIds=[1])</code>
<code>.loadAnns(ids)</code>	加载ID对应的数据	<code>train_coco.loadAnns([1,2])</code>
<code>.annToMask(annotation)</code>	将标注转换为掩码	
<code>.showAnns(annotations)</code>	使用matplotlib显示标注	

Focal Loss

来自kaiming

数据集

数据集

数据集 $L_i = -[y_i \log\{p_i\} + (1 - y_i) \log\{(1 - p_i)\}]$ 数据集 98 数据集 0.5 数据集 $L_{\{i\}} = -2 \log\{(0.5)\} = 1.386$ $L_{\{i\}} = 98 \times 0.683 = 67.914$ $L = \frac{1.386 + 67.914}{100} = 0.693$ 数据集 0.1 数据集 $L = \frac{2 \times 2.302 + 98 \times 0.105}{100} = 0.325$ 数据集

Focal Loss

数据集 $L = -\alpha_t (1 - p_t)^{\gamma} \log\{p_t\}$ 数据集

- α 数据集
- p_t 数据集 $1 - p_t$ 数据集
- γ 数据集
- 数据集