多任务学习

Multi-task Learning

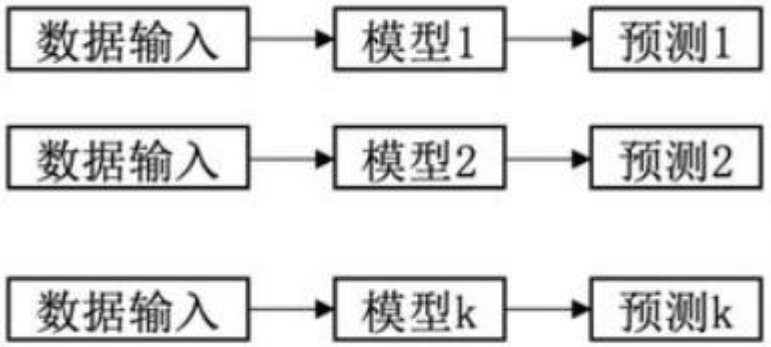
汇报人：孔金迪　汇报时间：2021年4月

目 录
Contents

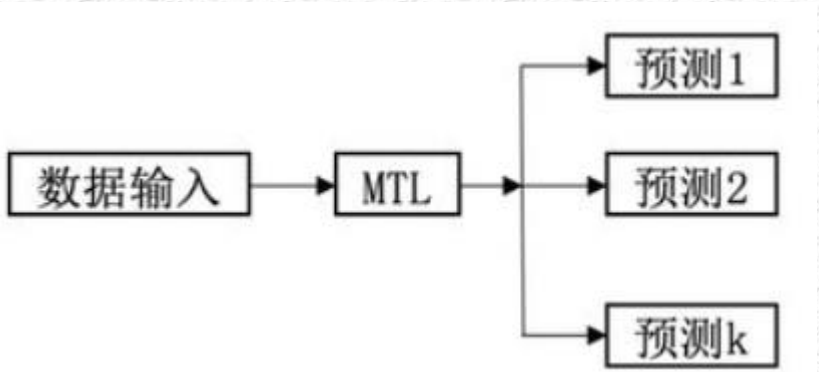
Part.01

多任务学习简介

1.1 多任务学习的定义



单任务学习



多任务学习

Guises:

- Joint Learning
- Learning to learn
- Learning with auxiliary tasks

多个损失函数

1.2 多任务学习的动机

- 方便！一次搞定多个任务
- 效果更好！缓解模型的过拟合，提高泛化能力
- 数据增强！不同任务的数据
- 任务互助！A、B搭配，干活不累~

Part.02

研究方向

MTL Network Design / What to share? 网络设计

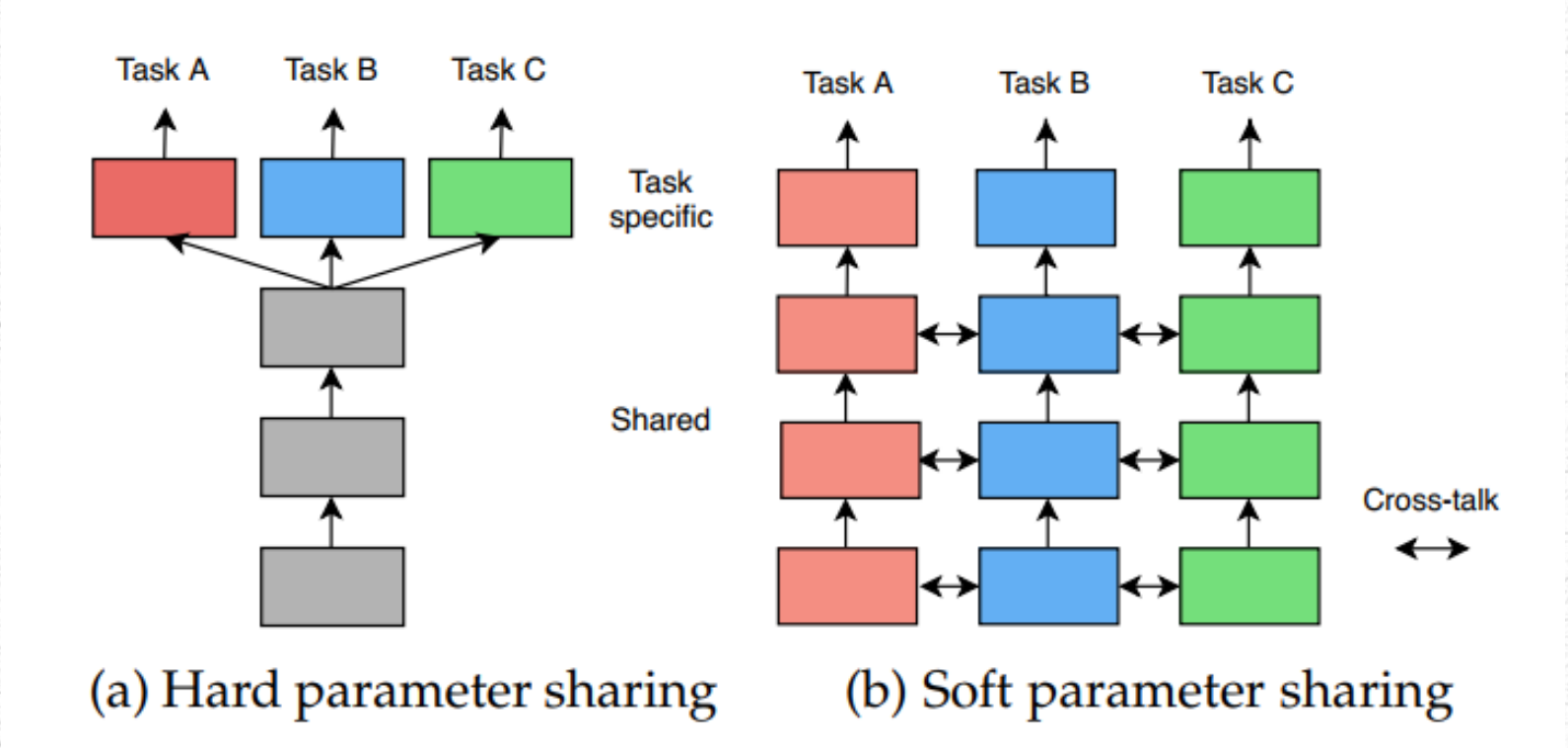
MTL Network Loss Function Design / How to learn? 损失函数设计与梯度优化

2.1 网络结构设计

Encoder-focused
Decoder-focused

- 特征如何共享?

2.1.1 特征如何共享？ (1)

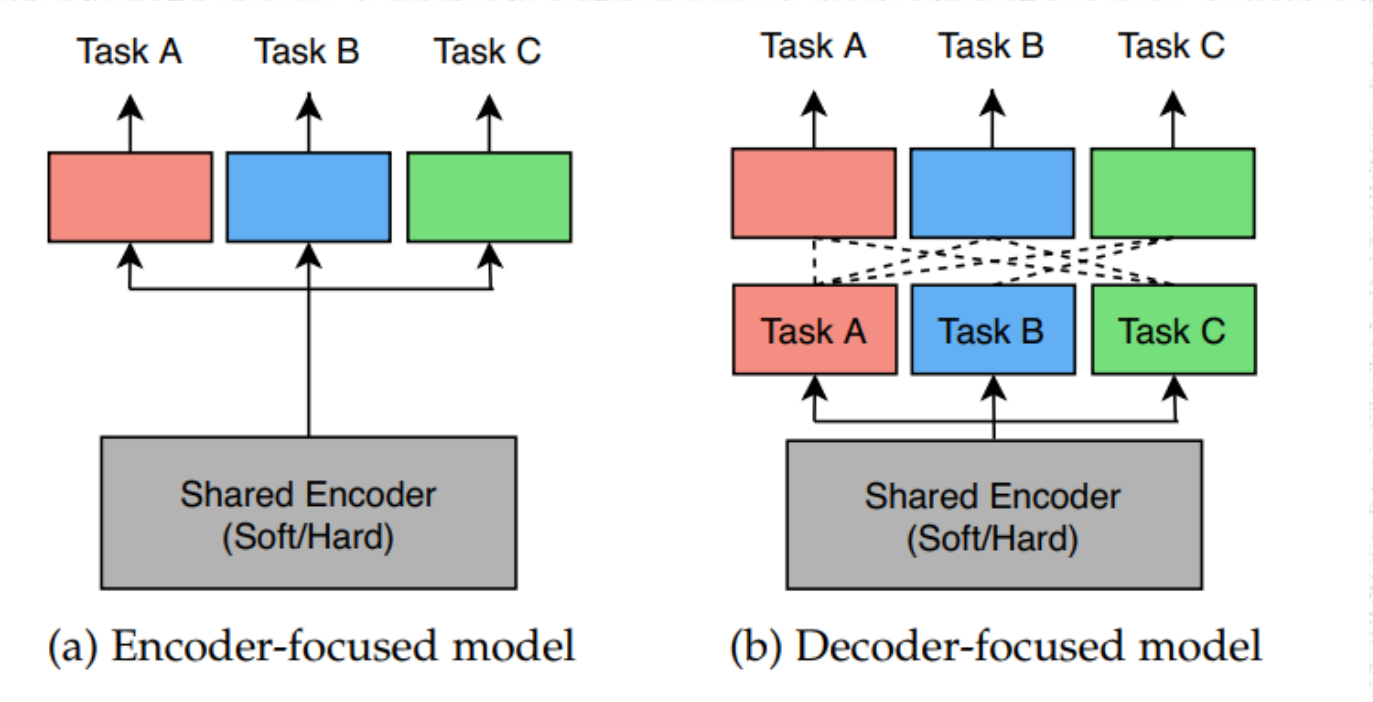


- 硬参数共享：
  - 一个共享骨干，共享参数
  - 任务特定(task-specific)的分支
  - 常作为baseline
- 有一些策略融合了上述两种共享方式

- 软参数共享：
  - 每个任务都具有自己的参数
  - Cross-task talk任务间交互

2.1.1 特征如何共享？ (2)

- 从任务交互/特征共享的位置进行区分
  - Encoder-focused model
  - Decoder-focused model

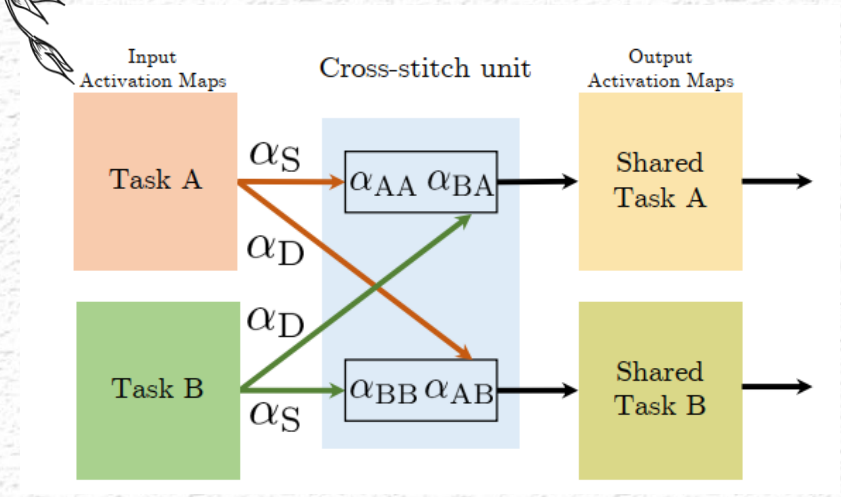


- 编码器为重点：
  - 编码器阶段共享特征
  - 任务特定(task-specific)头
  - 依赖编码器学习通用表示

- 解码器为重点：
  - 解码器阶段共享特征

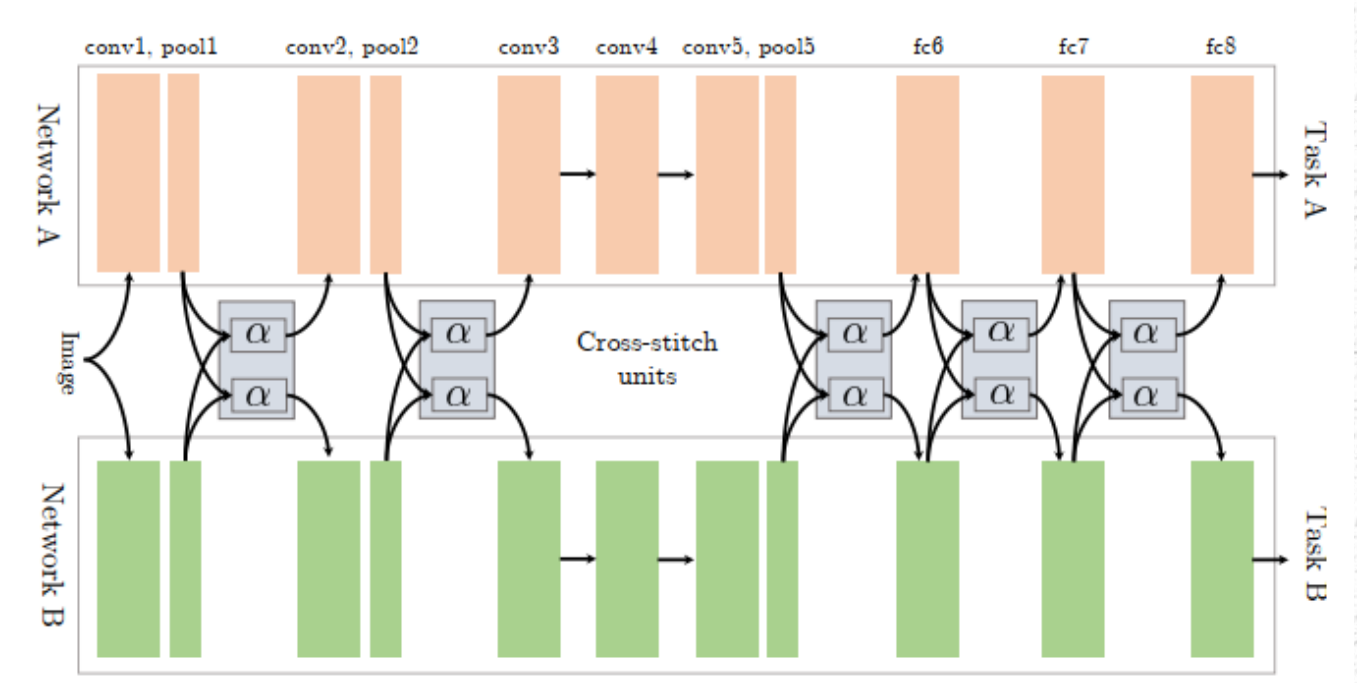
2.1.1 Encoder-focused

2.1.1 Encoder-focused : Cross-stitch Network



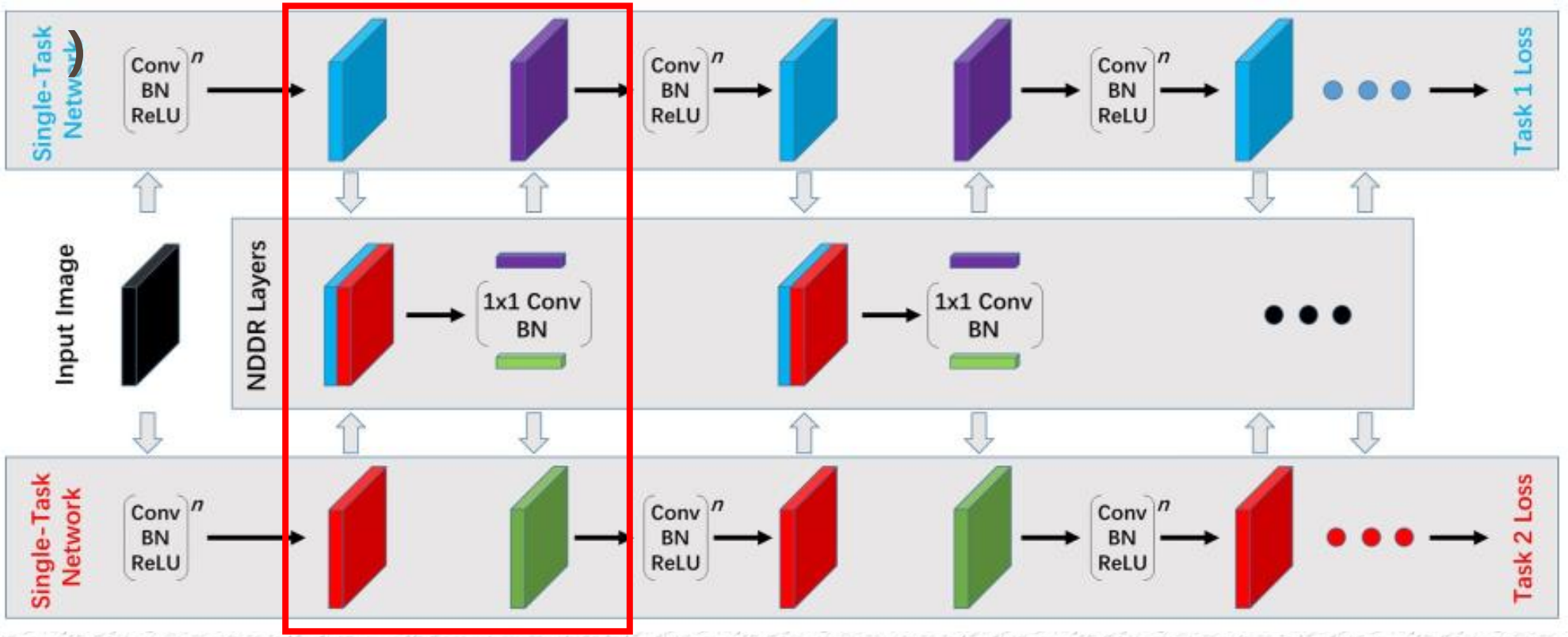
$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix}$$

若α_{BA} =0，则A任务不获取B的特征图，即不需要B的权重



- 在编码器的每层的输出和下一层输入间插入stitch unit，将上一层任务中 A&B的特征图进行线性组合
- 通过学习线性组合的权重，得出哪些层需要共享权重，哪些层不要贡献权值

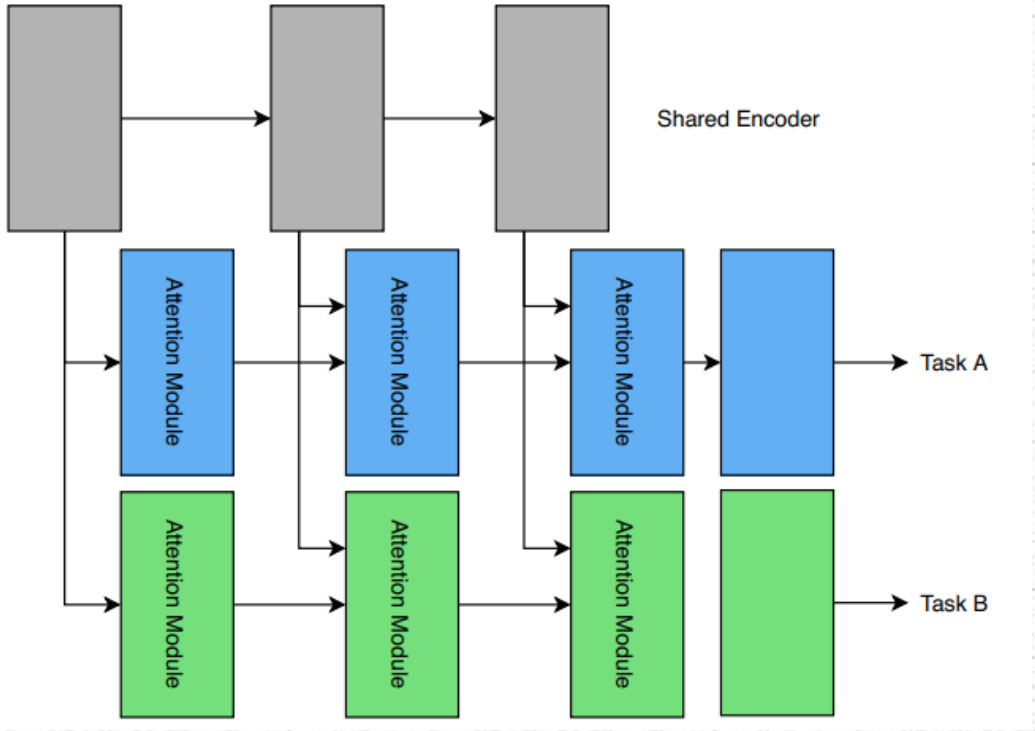
2.1.1 Encoder-focused : NDDR-CNNs(Neural Discriminative Dimensionality Reduction CNNs

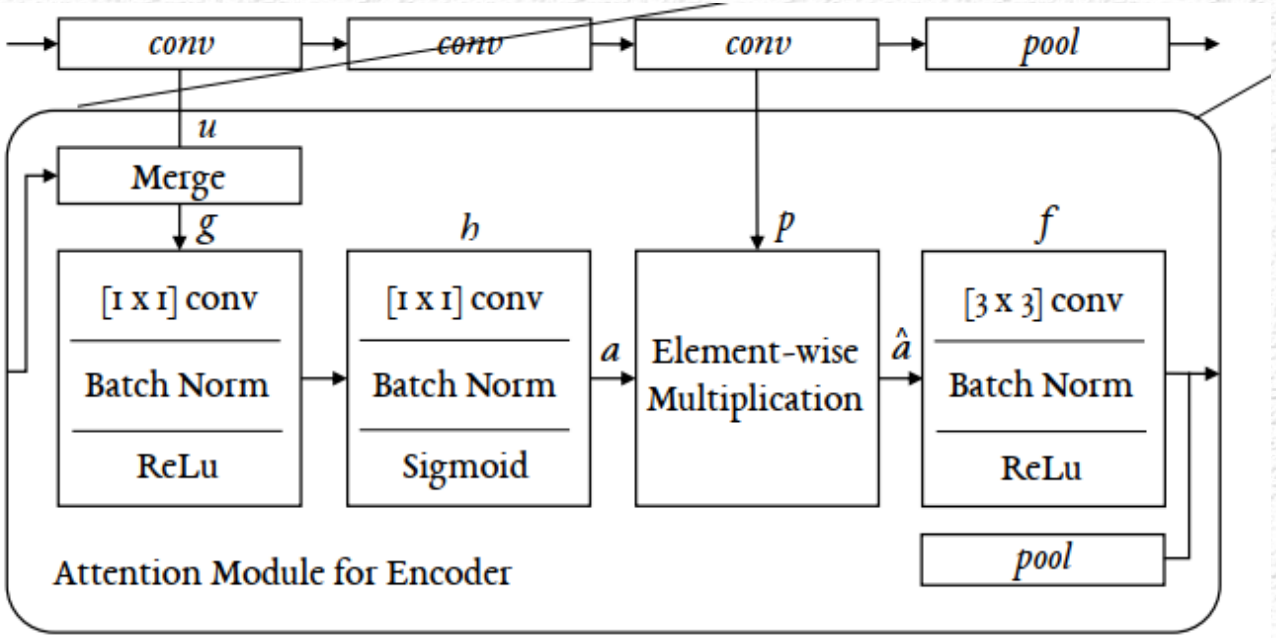


- NDDR层从两个单任务网络中获取特征
- 特征拼接后，1*1卷积降维

[1]Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L. Yuille. NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019

2.1.1 Encoder-focused : Multi-Task Attention Network





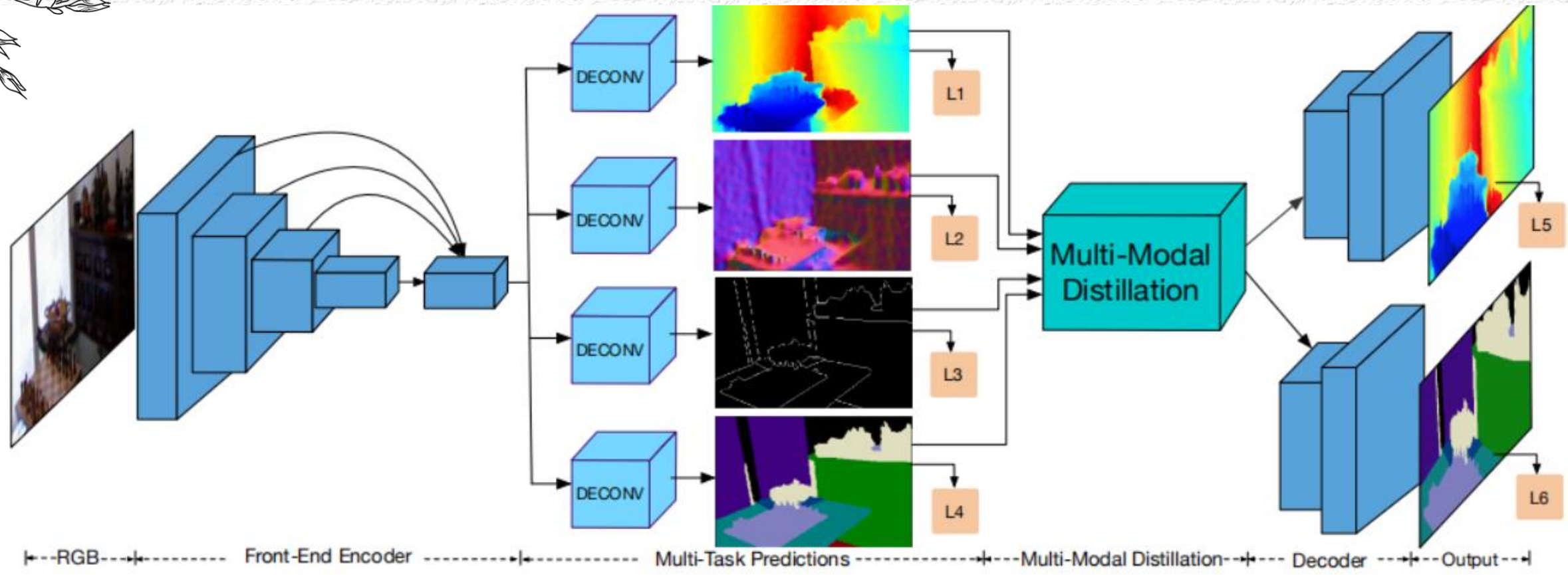
- 共享编码器（backbone）提取通用特征
- 每个任务特定的注意力模块从backbone中选择特征
- 注意力模块由Conv + Sigmoid实现

2.1.2 Decoder-focused

- Encoder-focused方法在一个处理周期中直接预测来自相同输入的所有任务输出，可能无法捕捉到任务之间的差异

- Decoder-focused方法：
  - 多任务网络进行初始预测
  - 利用初始预测进一步改进每个任务的输出
  - 解码器阶段共享或者交换信息

2.1.2 Decoder-focused : PAD-Net



$$F_k^o = F_k^i + \sum_{l \neq k} \sigma\left(W_{k,l} F_l^i\right) \odot F_l^i,$$

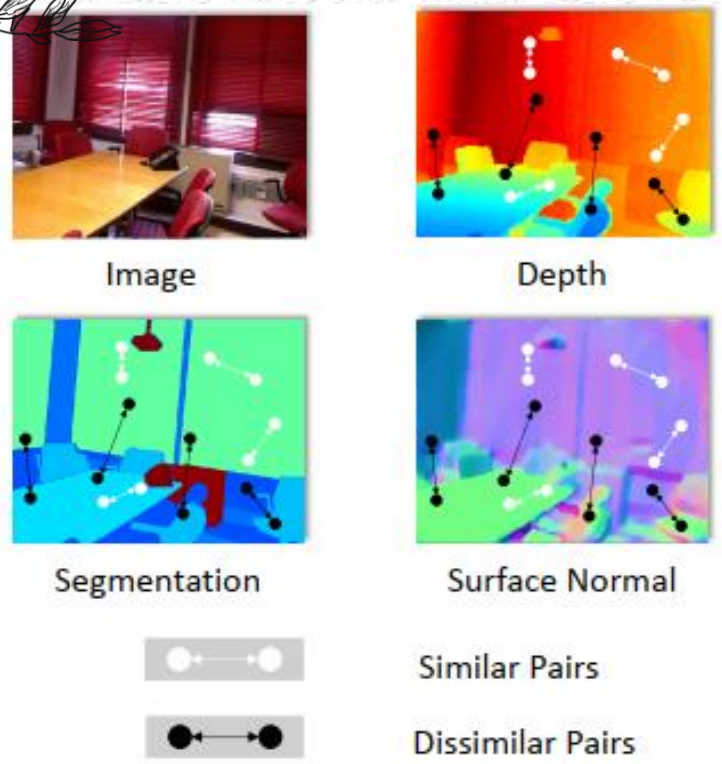
$\sigma\left(W_{k,l} F_l^i\right)$ **Spatial Attention Mask**

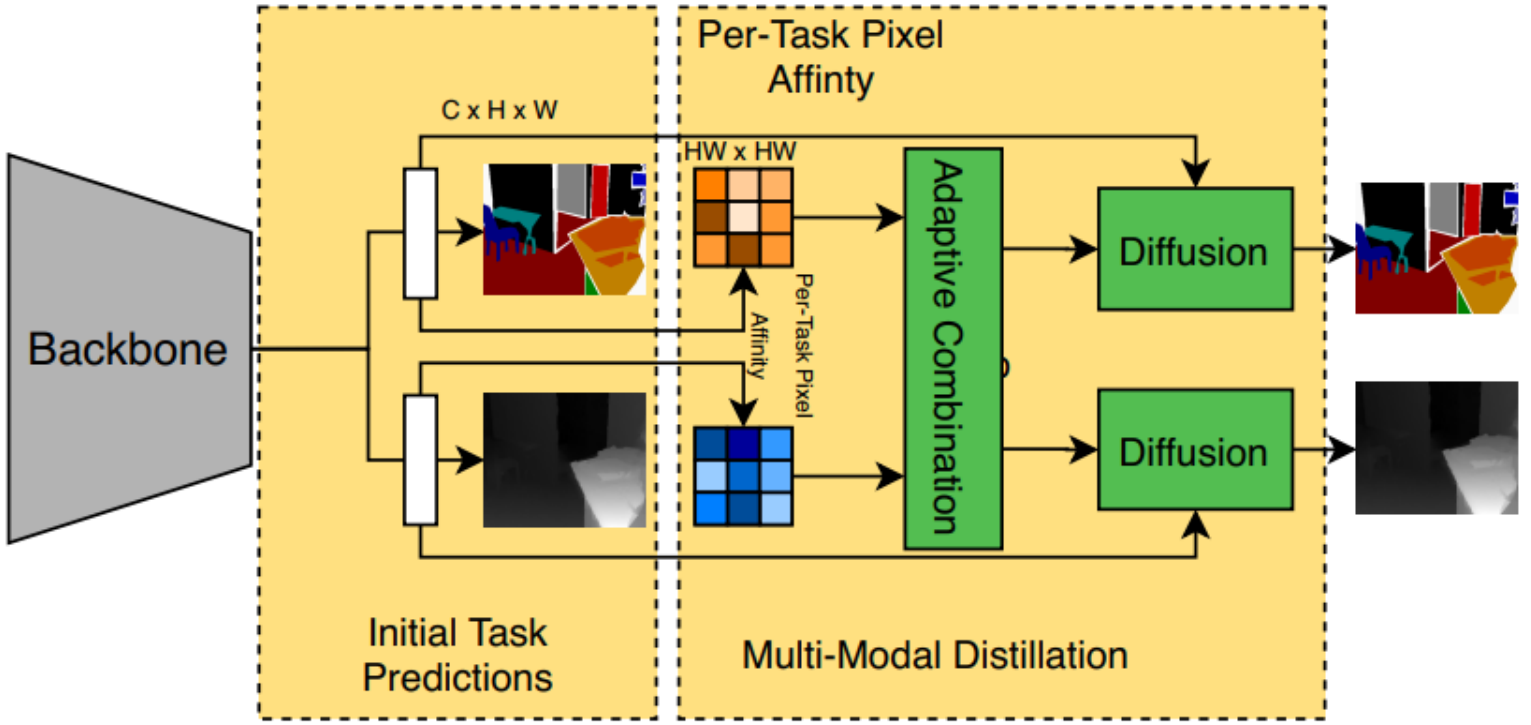
F_l^i **初始特征**

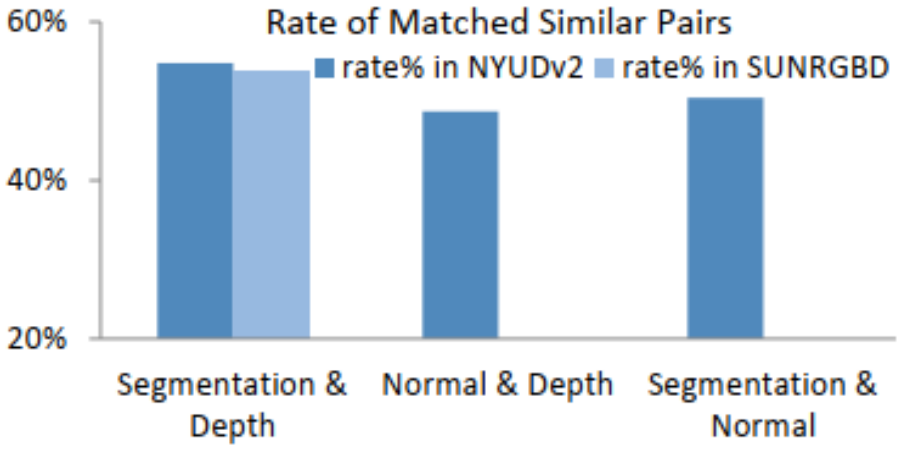
$l.$ **第$l$th个任务**

- Backbone提取的特征，用于做出初步预测
- 来自不同头部的特征通过distillation，做出最终预测
- 该框架中可以使用辅助任务（只生成初始预测，而不生成最终预测的任务）

2.2.2 Decoder-focused : PAP-Net



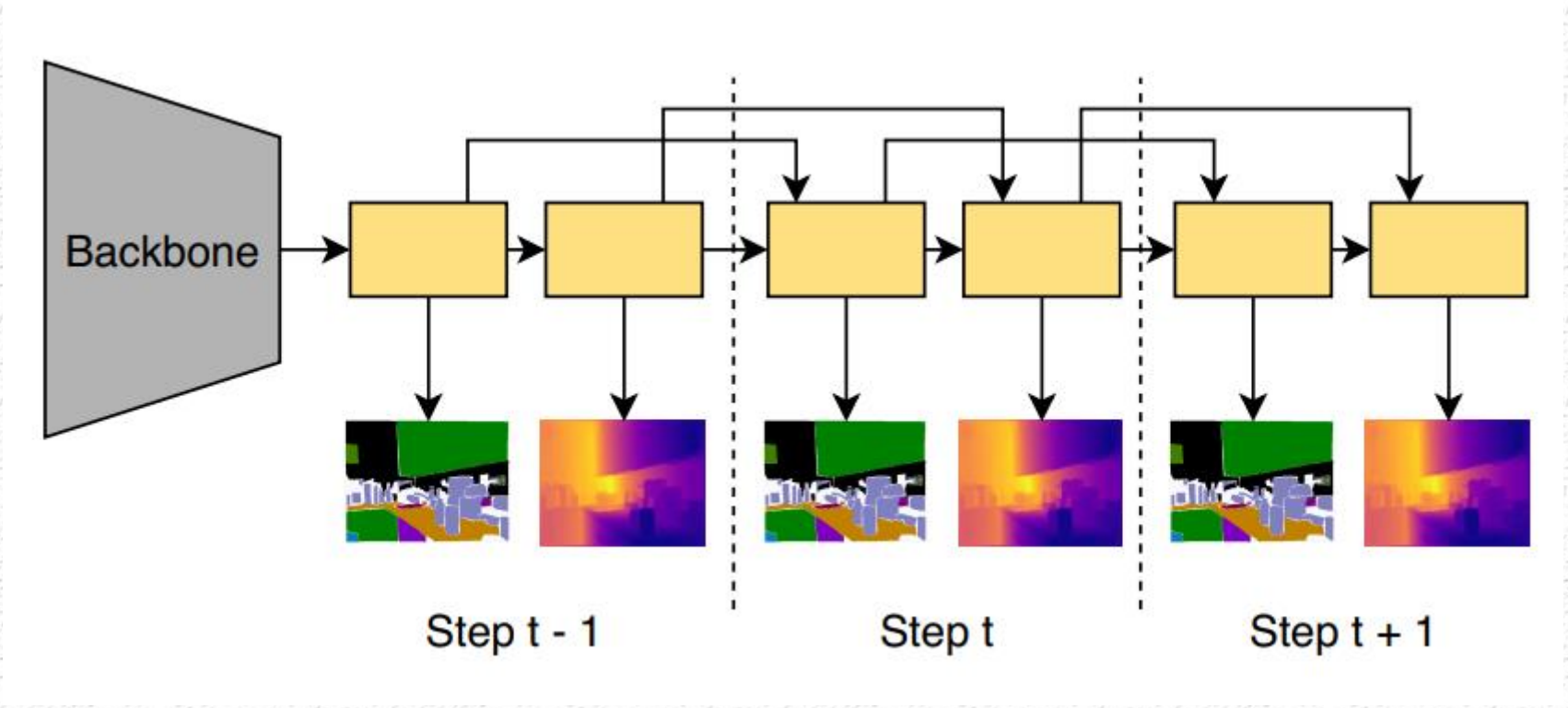




- 不同任务间一致的相似对占比较高
- 通过像素亲和性(pixel affinity)评估整幅图像上的Non-local像素关系

$$\hat{M}_{T_j} = \sum_{T_i} \alpha_i^{T_j} \cdot M_{T_i}.$$

2.2.2 Decoder-focused : JTRL(Joint Task-Recursive Learning)



- 任务在之前的基础上逐步交织在一起，而不是并行地预测
- 不能直接扩展到两个以上的任务中

2.2 优化策略

Task Balancing

- 避免出现某些任务占据主导地位

2.2.1 如何避免一枝独秀?

$$\mathcal{L}_{MTL} = \sum_i w_i \cdot \mathcal{L}_i.$$

$$W_{sh} = W_{sh} - \gamma \sum_i w_i \frac{\partial \mathcal{L}_i}{\partial W_{sh}}$$

- W_{sh}的优化受到所有loss的影响
- 网络中每个任务的影响是可控的:
  - 调节权重w_i
  - 调节梯度$\frac{\partial L_i}{\partial W_{sh}}$

w_i 每个任务的权重

$\mathcal{L}_i$, 每个任务的损失

W_{sh} 共享层的权重

2.2.2调节w_i：Uncertainty Weighting（1）

贝叶斯建模中的不确定性

- 认知不确定性：模型本身的不确定，因为缺乏训练数据，模型的认知不足，可以通过扩充训练集解决
- 偶然不确定性：数据本身存在的偏差，如出现了比较大的标注误差
  - 数据依赖（异方差）不确定性：依赖于输入数据的不确定性，体现在模型的输出上
  - 任务依赖（同方差）不确定性：不依赖于输入数据，也不是模型输出结果；而是对所有输入数据相同的常量，对不同任务不同的变量

↑

不同任务间的相对难度

↑

最大化

2.2.2调节w_i：Uncertainty Weighting（2）

回归 $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \mathcal{N}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma^2)$

分类 $p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = \text{Softmax}(\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$

$$p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \cdot p(\mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$
$$= \mathcal{N}(\mathbf{y}_1; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_1^2) \cdot \mathcal{N}(\mathbf{y}_2; \mathbf{f}^{\mathbf{W}}(\mathbf{x}), \sigma_2^2)$$

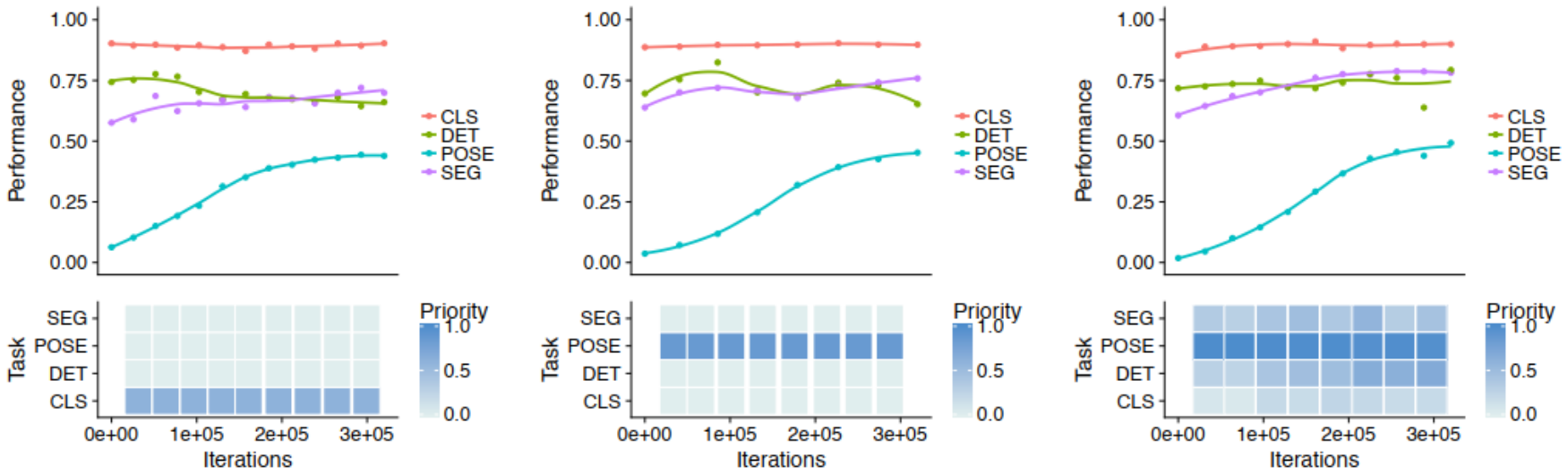
$$p(\mathbf{y}_1, ..., \mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) = p(\mathbf{y}_1|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))...p(\mathbf{y}_K|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$

$$= -\log p(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}^{\mathbf{W}}(\mathbf{x}))$$
$$\propto \frac{1}{2\sigma_1^2}||\mathbf{y}_1 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \frac{1}{2\sigma_2^2}||\mathbf{y}_2 - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 + \log \sigma_1\sigma_2$$
$$= \frac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \frac{1}{2\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log \sigma_1\sigma_2$$

$$\log p(\mathbf{y}|\mathbf{f}^{\mathbf{W}}(\mathbf{x})) \propto -\frac{1}{2\sigma^2}||\mathbf{y} - \mathbf{f}^{\mathbf{W}}(\mathbf{x})||^2 - \log \sigma$$

$$\mathcal{L}(W, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2}\mathcal{L}_1(W) + \frac{1}{2\sigma_2^2}\mathcal{L}_2(W) + \log \sigma_1\sigma_2$$

2.2.2调节w_i： Dynamic Task Prioritization



(a) Prioritize Easy (Fixed) **(b)** Prioritize Hard (Fixed) **(c) Dynamic (Ours)**

$$\mathrm{CE}(p_c) = -\log(p_c) \qquad \text{where} \qquad p_c = \begin{cases} p, & \text{if } y = 1 \\ 1-p, & \text{otherwise} \end{cases}$$

$$\mathrm{FL}(p_c;\gamma_0) = -(1-p_c)^{\gamma_0}\log(p_c)$$

$$\mathcal{L}_t^*(\cdot) = \mathrm{FL}(p_c;\gamma_0)$$

$$\mathcal{L}_{\mathrm{DTP}}(\cdot) = \mathcal{L}^*_{\mathrm{Total}}(\cdot) = \sum_{t=1}^{|T|} \mathrm{FL}(\bar{\kappa}_t;\gamma_t)\mathcal{L}_t^*(\cdot)$$

- 不让学习资源一直倾斜给最简单/最困难的
- 根据不同任务难度分配不同权重，将困难任务的优先级排在前面

2.2.2调节 $\frac{\partial L_i}{\partial W_{sh}}$：Multiple Gradient Descent Algorithm (MGDA)

- MTL 的另一个目标是找到不受任何其它任务主导的解决方案。据说这种方案就是帕累托最优（Pareto Optimal），即不能在不损失其他目标的情况下优化一个目标
- 寻找帕累托最优解的问题也被称为多目标优化。目前已有多种多目标优化算法，其中一种叫多梯度下降算法（MGDA）

Algorithm 2 Update Equations for MTL

for $t = 1$ **to** T **do**
  $\boldsymbol{\theta}^t = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}^t} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on task-specific parameters
end for
$\alpha^1, \ldots, \alpha^T = \text{FRANKWOLFESOLVER}(\boldsymbol{\theta})$ ▷ Solve (3) to find a common descent direction
$\boldsymbol{\theta}^{sh} = \boldsymbol{\theta}^{sh} - \eta \sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on shared parameters

procedure FRANKWOLFESOLVER($\boldsymbol{\theta}$)
  Initialize $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^T) = (\frac{1}{T}, \ldots, \frac{1}{T})$
  Precompute $\mathbf{M}$ st. $\mathbf{M}_{i,j} = \left(\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^i(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^i)\right)^\intercal \left(\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^j(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^j)\right)$
  repeat
    $\hat{t} = \arg\min_r \sum_t \alpha^t \mathbf{M}_{rt}$
    $\hat{\gamma} = \arg\min_\gamma \left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)^\intercal \mathbf{M} \left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)$ ▷ Using Algorithm 1
    $\boldsymbol{\alpha} = (1-\hat{\gamma})\boldsymbol{\alpha} + \hat{\gamma} \boldsymbol{e}_{\hat{t}}$
  until $\hat{\gamma} \sim 0$ **or** Number of Iterations Limit
  return $\alpha^1, \ldots, \alpha^T$
end procedure

Algorithm 1
$\min_{\gamma \in [0,1]} \|\gamma\boldsymbol{\theta} + (1-\gamma)\bar{\boldsymbol{\theta}}\|_2^2$

if $\boldsymbol{\theta}^\intercal \bar{\boldsymbol{\theta}} \geq \boldsymbol{\theta}^\intercal \boldsymbol{\theta}$ **then**
  $\gamma = 1$
else if $\boldsymbol{\theta}^\intercal \bar{\boldsymbol{\theta}} \geq \bar{\boldsymbol{\theta}}^\intercal \bar{\boldsymbol{\theta}}$ **then**
  $\gamma = 0$
else
  $\gamma = \frac{(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta})^\intercal \bar{\boldsymbol{\theta}}}{\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|_2^2}$
end if

2.3 哪种方法更有效

2.3 哪种方法更有效

- 网络架构：
  - Encoder-focused:
    - 更注重性能时，选择Cross-Stitch
    - 运算资源受限时，选择Multi-task Attention Network
  - Decoder-focused:
    - 这种结构的结果相对来说更令人满意
    - PAP-Net 和 JTRL在FLOPS（运算速度）上面提升多
- 优化策略
    - 网格化搜索各任务的权值，线性加权，效果优于其他方法

Part.03 总结

3.1 重新思考多任务学习的动机

- 方便！一次搞定多个任务
  - 主任务 & 辅任务
  - 场景理解：语义标签预测（分割） & 深度估计 & 表面法线预测

- 效果更好！缓解模型的过拟合，提高泛化能力
  - MTL是归纳迁移(Inductive Transfer)的一种
  - 归纳迁移通过归纳偏好(Inductive Bias)来改善模型，提高泛化能力
  - 辅助任务提供了归纳偏好

- 任务互助！
  - 模型可以学习更复杂的任务
  - 把已经获得的知识用于相关学习任务中
    - 共享的特征

某瓜：(色泽=青绿；根蒂=蜷缩；敲声=沉闷)

假设1：好瓜 ⟷ (色泽=青绿；根蒂=蜷缩；敲声=沉闷)

假设2：好瓜 ⟷ (色泽=*；根蒂=蜷缩；敲声=浊响)

谢谢观看

Thank you for watching