

Lane Segmentation

主要内容

- 构建项目Baseline
- 训练策略

学习目标

- 掌握Baseline结构和实现
- 掌握语义分割的Metrics
- 掌握语义分割的Loss
- 掌握训练策略

Baseline

- 训练
- 推断
- 配置

Baseline原则

- no data augmentation
- no **big** model
- low resolution (**先做缩小**)
- little tricks

训练

1. 创建网络模型
2. 加载预训练权重（可选）
3. 设置优化器
4. 生成训练数据

训练

1. epoch/batchnum(iterations)
2. 调整学习率
3. 前向计算 ($\text{pred} = \text{model}(\text{input})$)
4. 计算损失 ($\text{loss} = \text{pred} - \text{label}$)
5. 反向传播

模型

- U-Net (Residual Block)
- 多个模型

The No Free Lunch Theorem

- Learning theory claims that a machine learning algorithm can generalize well from a finite training set of examples. This seems to contradict some basic principles of logic.

The No Free Lunch Theorem

- no machine learning algorithm is universally any better than any other

The No Free Lunch Theorem

- If we make assumptions about the kinds of **probability distributions** we encounter in real-world applications, then we can design learning algorithms that perform well on these distributions.

SGD+Momentum

- 手动挡
- 对学习率等超参数敏感

Adam

- 自动挡
- 最常用

数据生成器

Learning Rate

- Weight Initialization
- BN
- Adam...

Learning Rate

- Decay
- Cycle
- Warmup

Metrics

- CityScape

<https://www.cityscapes-dataset.com/benchmarks/>

TP/FP/FN/TN

我们可以使用一个 2x2 [混淆矩阵](#)来总结我们的“狼预测”模型，该矩阵描述了所有可能出现的结果（共四种）：

真正例 (TP):

- 真实情况：受到狼的威胁。
- 牧童说：“狼来了。”
- 结果：牧童是个英雄。

假正例 (FP):

- 真实情况：没受到狼的威胁。
- 牧童说：“狼来了。”
- 结果：村民们因牧童吵醒他们而感到非常生气。

假负例 (FN):

- 真实情况：受到狼的威胁。
- 牧童说：“没有狼”。
- 结果：狼吃掉了所有的羊。

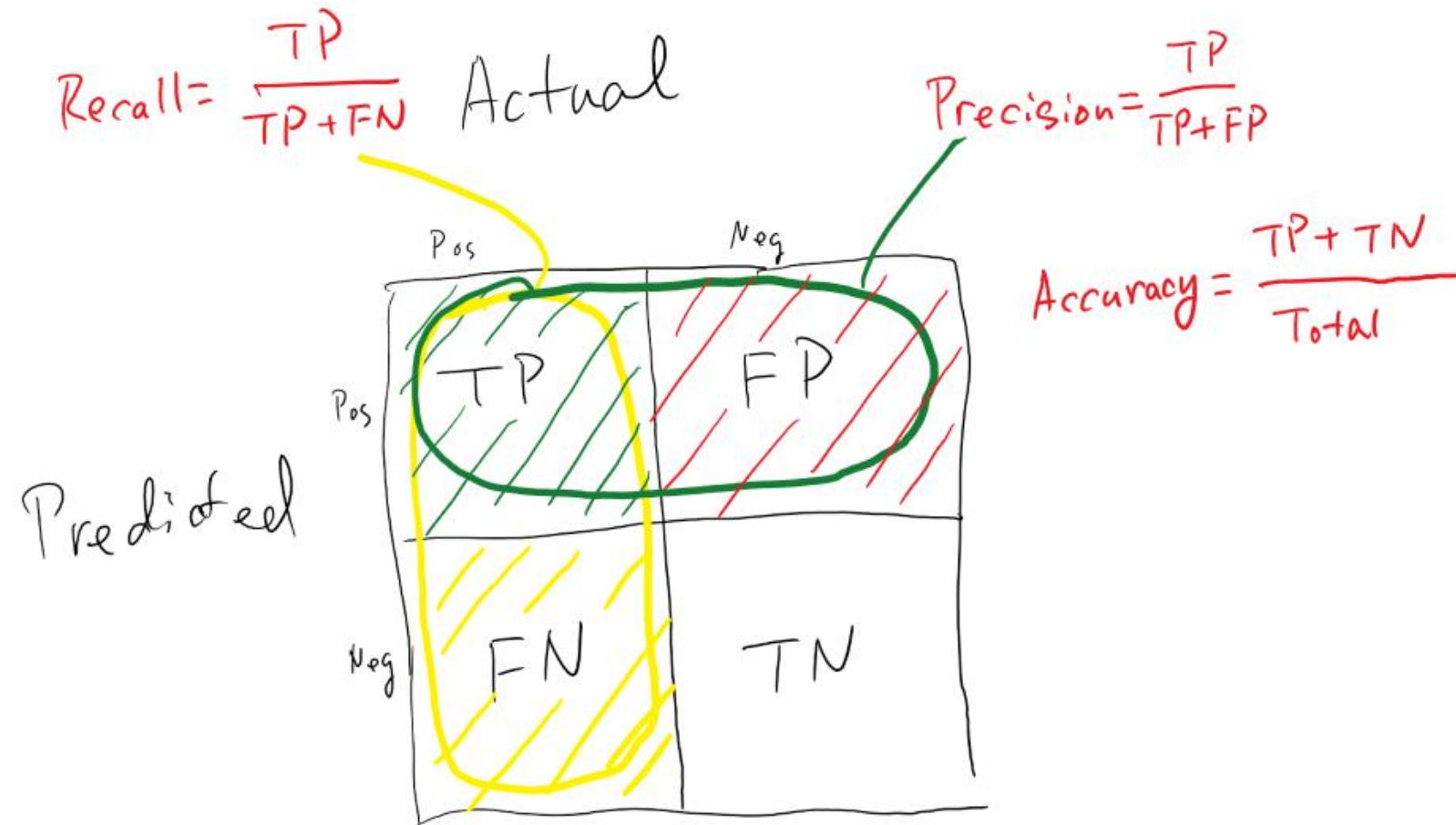
真负例 (TN):

- 真实情况：没受到狼的威胁。
- 牧童说：“没有狼”。
- 结果：大家都没事。

真正例是指模型将正类别样本正确地预测为正类别。同样，**真负例**是指模型将负类别样本正确地预测为负类别。

假正例是指模型将负类别样本错误地预测为正类别，而**假负例**是指模型将正类别样本错误地预测为负类别。

Precision/Recall = $\frac{1}{2}$



confusion matrix

by 吳

gt

True Class

airplane	923	4	21	8	4	1	5	5	23	6
automobile	5	972	2					1	5	15
bird	26	2	892	30	13	8	17	5	4	3
cat	12	4	32	826	24	48	30	12	5	7
deer	5	1	28	24	898	13	14	14	2	1
dog	7	2	28	111	18	801	13	17		3
frog	5		16	27	3	4	943	1	1	
horse	9	1	14	13	22	17	3	915	2	4
ship	37	10	4	4		1	2	1	931	10
truck	20	39	3	3			2	1	9	923

airplane
automobile

bird

cat

deer

dog

frog

horse

ship

truck

Predicted Class

pred

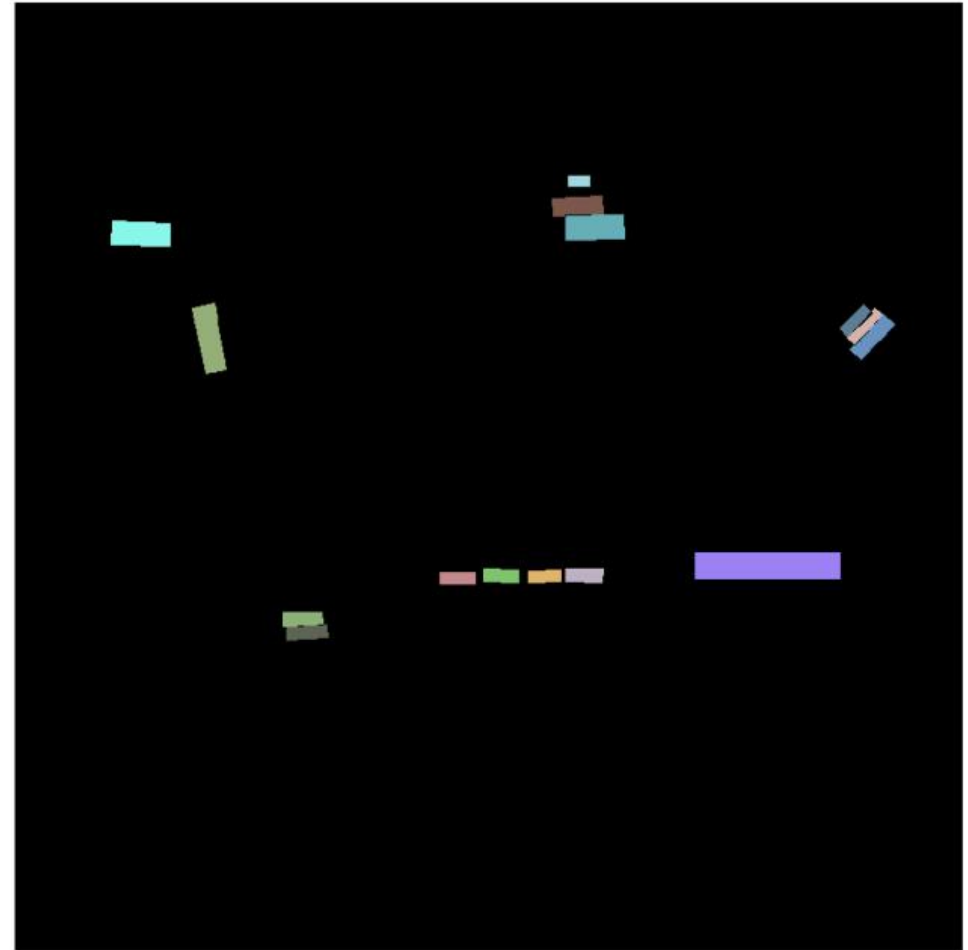
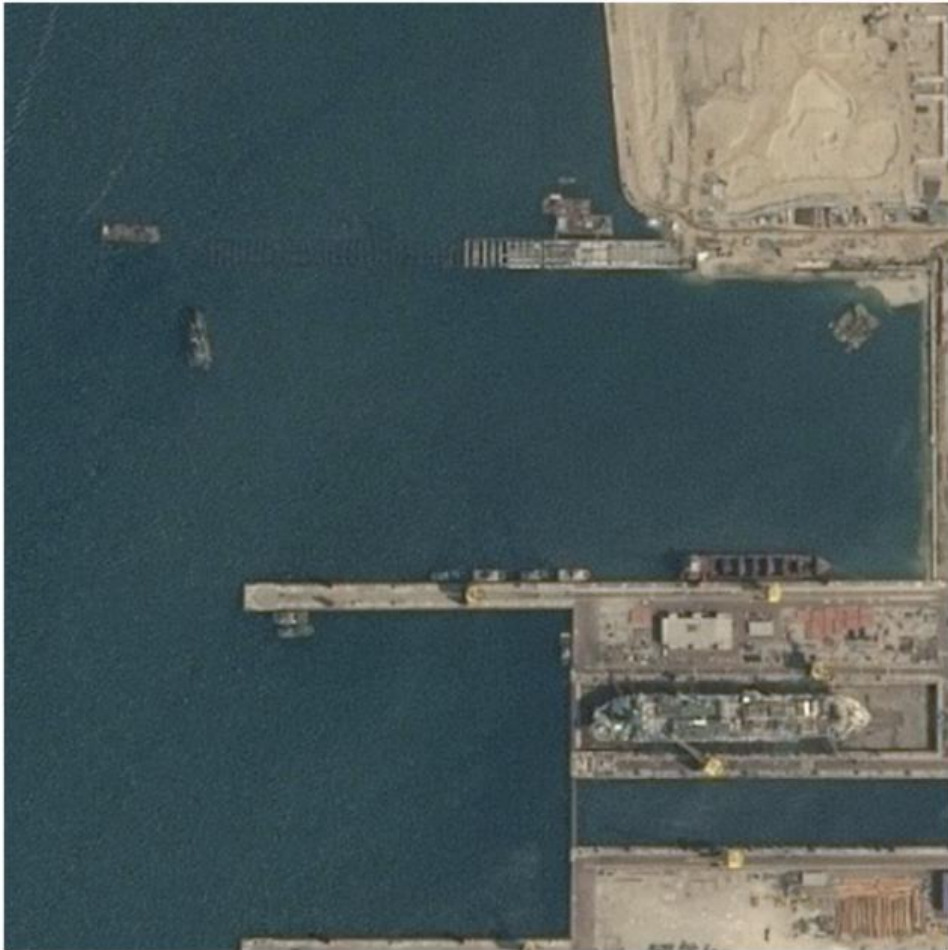
nx gt + pred

F-Score

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 score is the harmonic mean of precision and recall. Values range from 0 (bad) to 1 (good).

Pixel Accuracy




Pixel Accuracy



class imbalance

- Unfortunately, class imbalance is prevalent in many real world data sets, so it can't be ignored. Therefore, I present to you two alternative metrics that are better at dealing with this issue

Jaccard Index/IoU

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


The diagram illustrates the Jaccard Index (IoU) calculation. It consists of two parts. The top part shows two overlapping squares: one with a blue outline and one with a solid blue fill. The bottom part shows the union of these two squares as a single solid blue shape.

mIoU

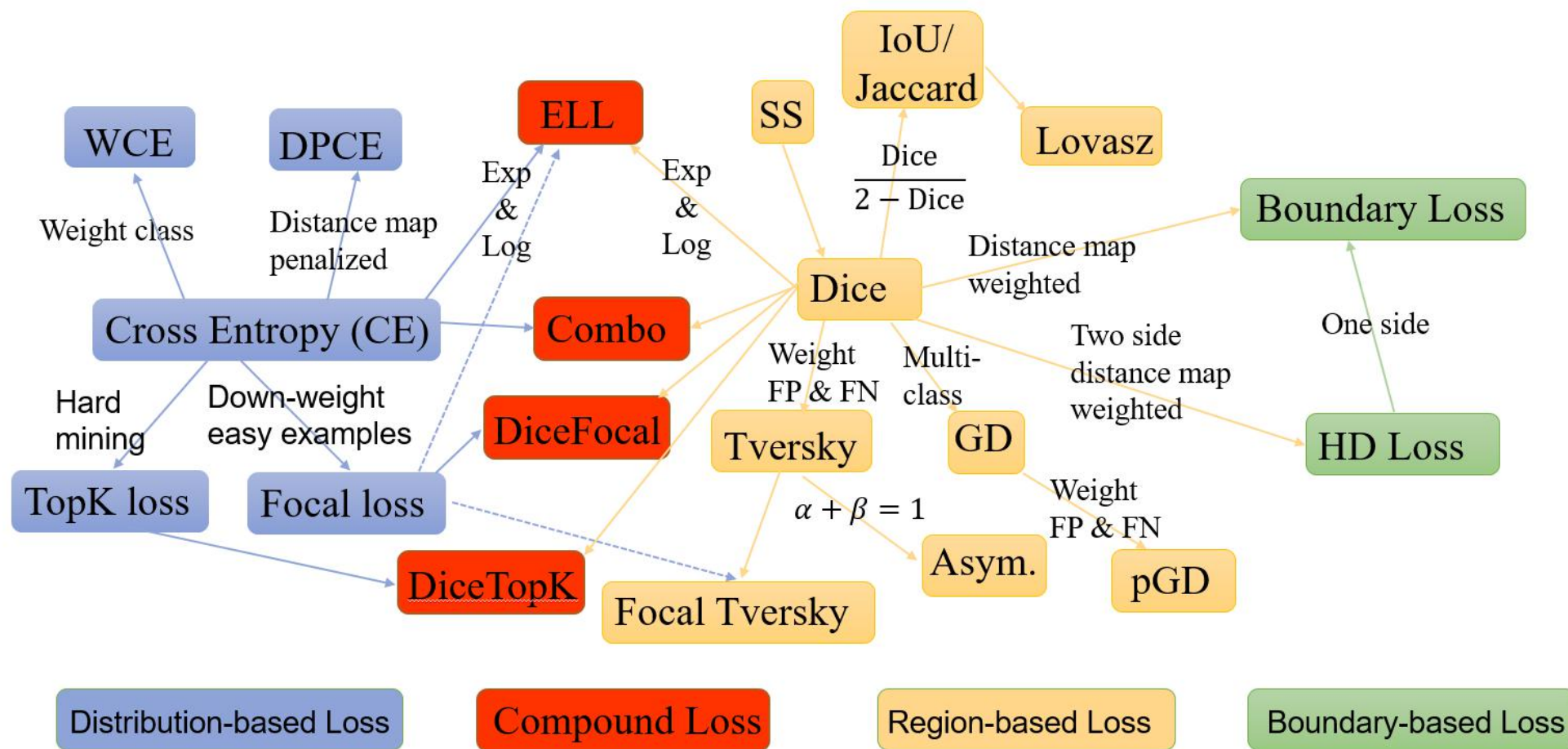
- For binary (two classes) or multi-class segmentation, the mean IoU of the image is calculated by taking the IoU of each class and averaging them.

IoU by bounding box

IoU by pixels

- 直方图--->混淆矩阵--->mIoU
- np.bincount
- np.diag

Loss



Loss

- A collection of loss functions for medical image segmentation
- <https://github.com/JunMa11/SegLoss>

Cross-Entropy

$$D_{KL}(p \parallel q) = \overset{\text{Entropy}}{H(p, q)} - \underset{\text{Cross entropy}}{H(p)}$$

ce loss

$$\textit{Cross — entropy loss} = - \underbrace{\sum_{c=1}^M Y \log(P)}$$

bce loss

$$-(y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

Information Theory

- Probability and Information Theory
- <http://www.deeplearningbook.org/contents/prob.html>

What is Information

- Probability
- frequentist probability

self-information

- Likely events should have low information content, and in the extreme case, events that are guaranteed to happen should have no information content whatsoever.
- Less likely events should have higher information content.
- Independent events should have additive information. For example, finding out that a tossed coin has come up as heads twice should convey twice as much information as finding out that a tossed coin has come up as head once.

self-information

$$I(x) = -\log P(x).$$

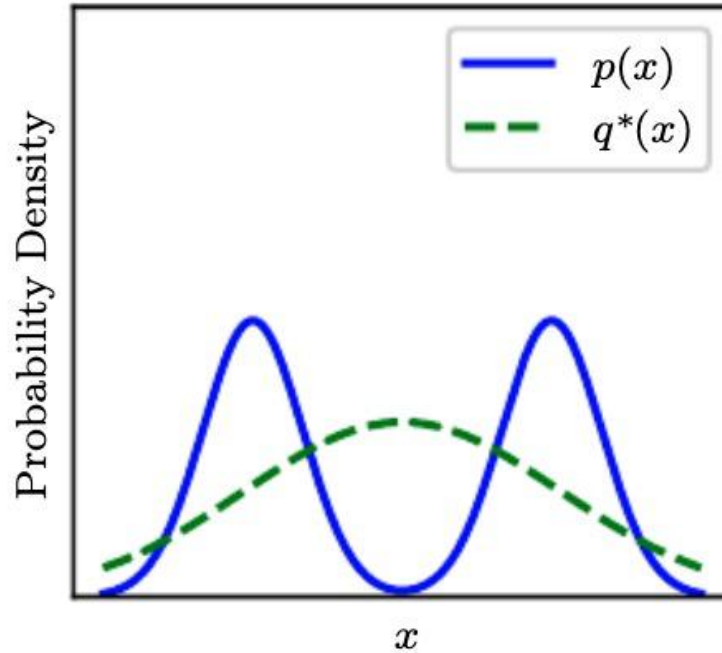
大道至简

Kullback–Leibler (KL) divergence

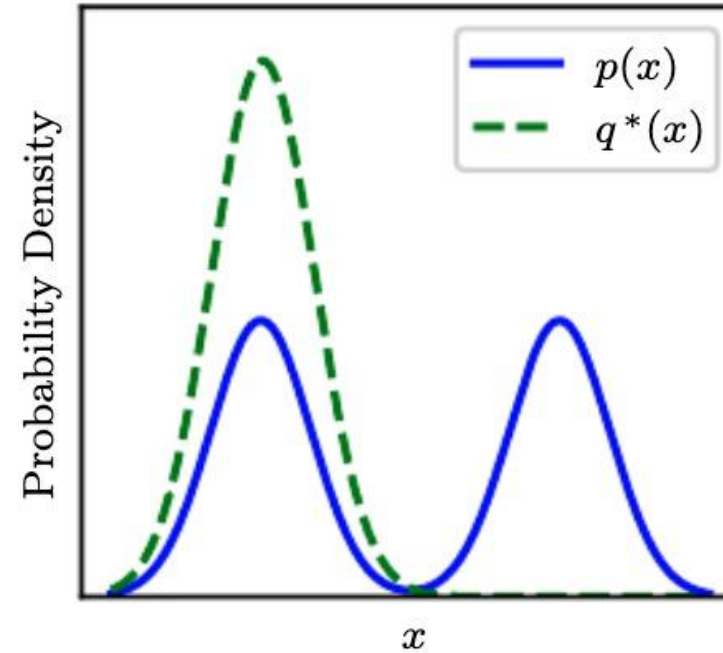
$$D_{\text{KL}}(P\|Q) = \mathbb{E}_{\mathbf{x} \sim P} \left[\log \frac{P(\mathbf{x})}{Q(\mathbf{x})} \right] = \mathbb{E}_{\mathbf{x} \sim P} [\log P(\mathbf{x}) - \log Q(\mathbf{x})] .$$

Kullback–Leibler (KL) divergence

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p \| q)$$



$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q \| p)$$



nll loss

- cross-entropy loss

CE loss by pixels

dice loss

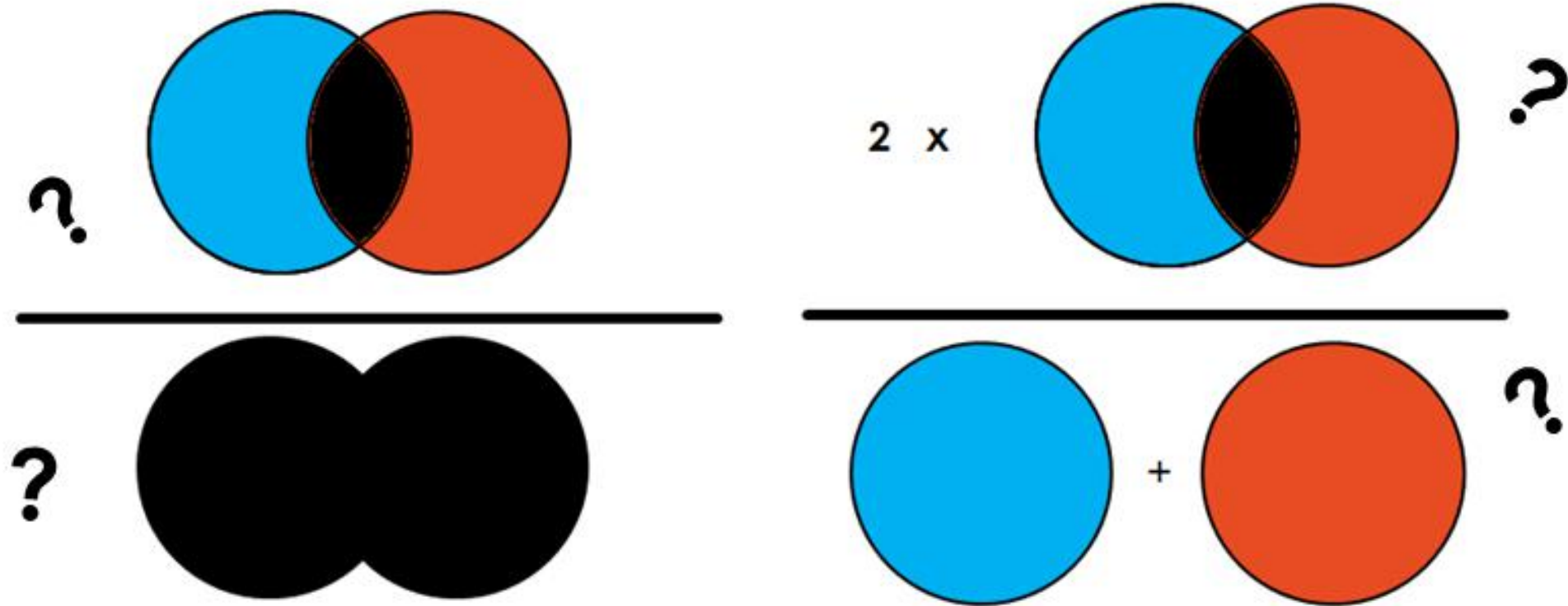
- V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation
- <https://arxiv.org/abs/1606.04797>

Combinations

$$\text{CE} (p, \hat{p}) + \text{DL} (p, \hat{p})$$

Note that CE returns a tensor, while DL returns a scalar for each image in the batch. This way we combine local (CE) with global information (DL).

IoU and Dice Coefficient



推断

1. 创建网络模型
2. 加载预训练权重
3. 生成推断数据
4. 推断（前向计算）
5. 后处理

配置

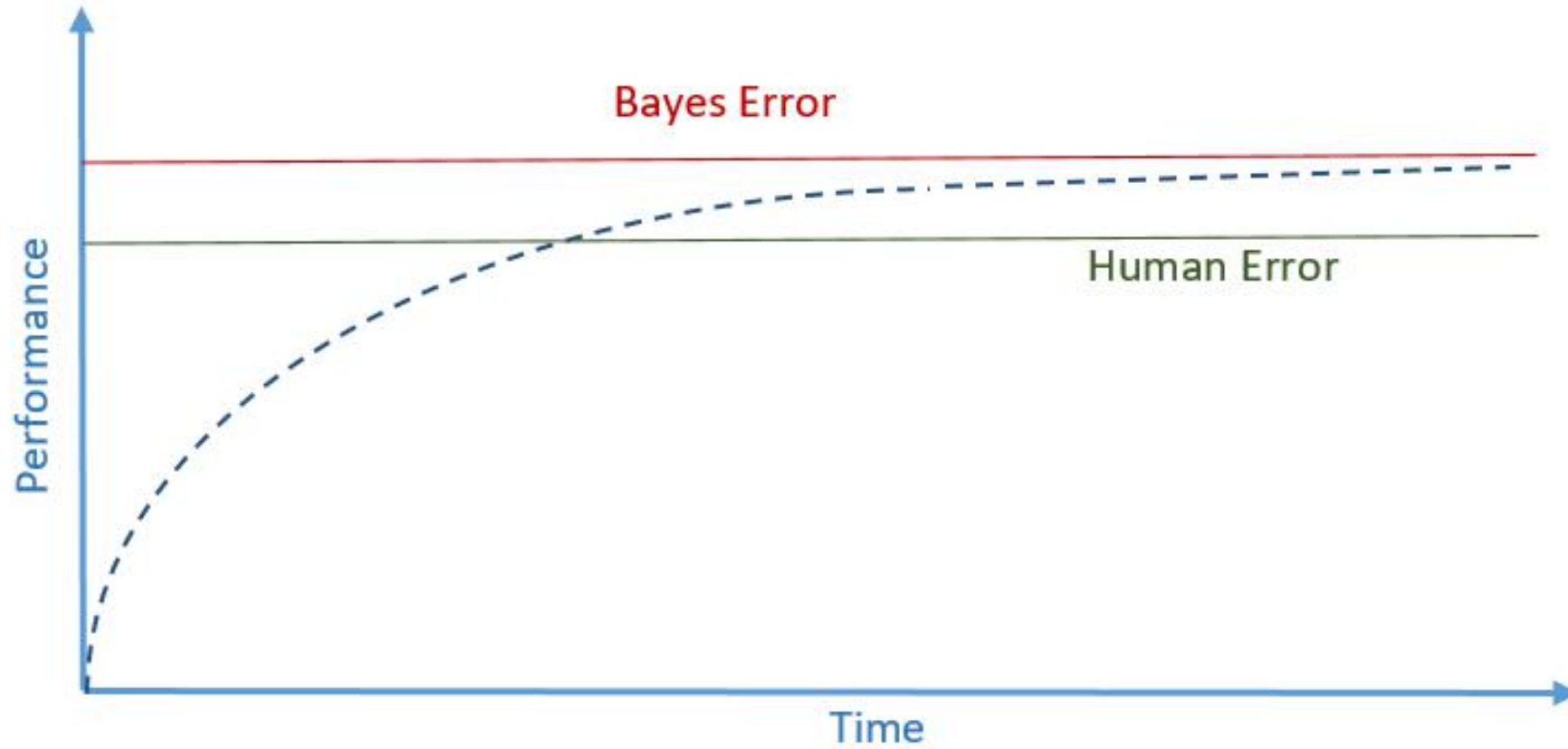
training strategy

- Collect more data
- Train algorithm longer
- Try Adam instead of gradient descent
- Add regularizations
- Network architecture
-

training strategy

- One of the challenges with building machine learning systems is that there's so **many things you could try**, so many things you could change.
- hyperparameters

Bayes Error



human-level performance

- define your key priorities
- based on **observations** of performances and dataset

Bias

- human-level performance
- Underfitting

Variance

- Overfitting

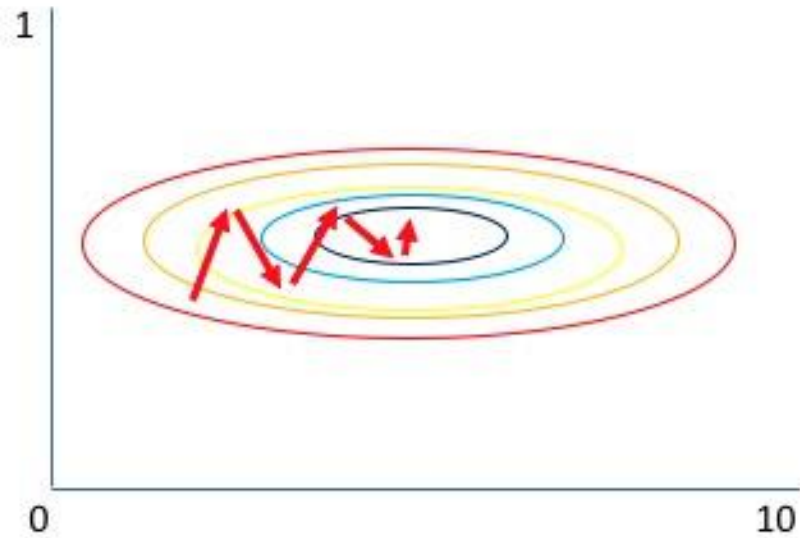
Training

- 详细观察数据（Bayes错误率）
- 确定Baseline
- 对Baseline模型进行优化（主要矛盾）
- ablation study

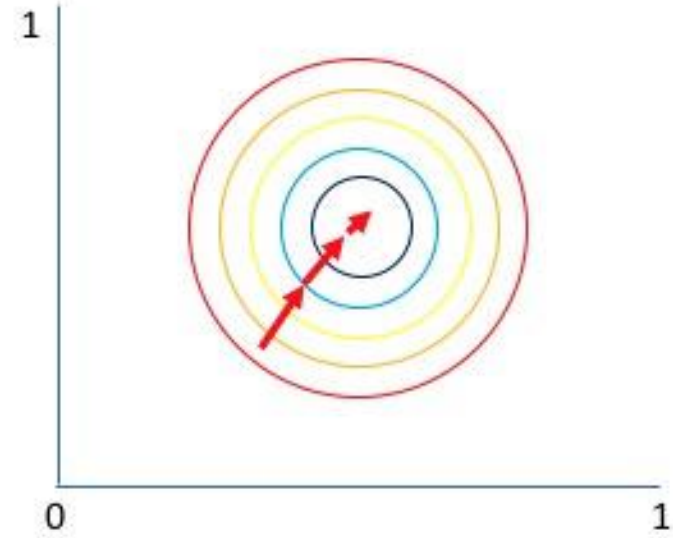
Orthogonalization



Normalization



Gradient of larger parameter
dominates the update



Both parameters can be
updated in equal proportions

Validation

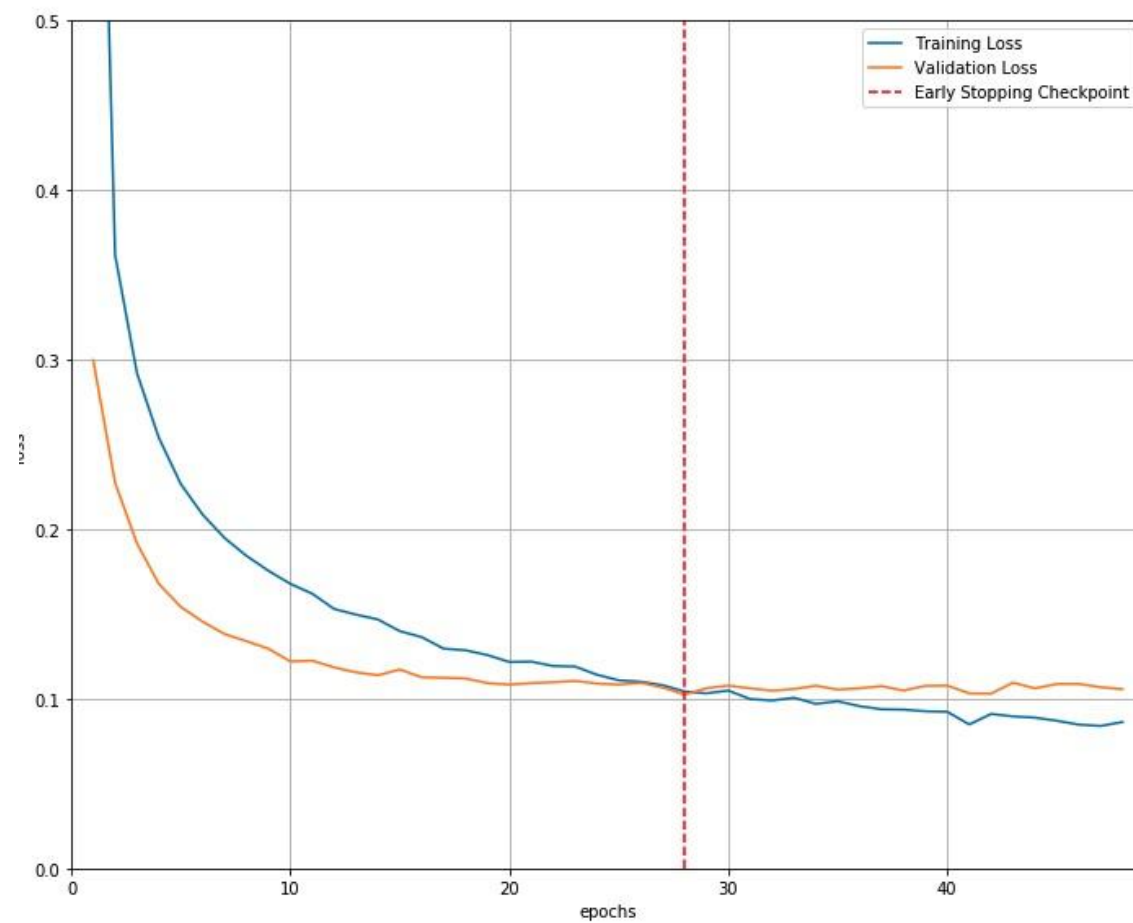
number of training epochs

- Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.

Early-Stop

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
# This callback will stop the training when there is no improvement in
# the validation loss for three consecutive epochs.
model.fit(data, labels, epochs=100, callbacks=[callback],
          validation_data=(val_data, val_labels))
```

Early-Stop



save model

- early stop

课程总结

- 项目Baseline的结构和实现

重难点

- Baseline的实现
- mIoU的计算

课程作业

- 以U-Net或deeplab作为模型实现项目Baseline的**训练代码**
 - Loss使用CE即可，不需要使用dice loss
 - 不需要计算Metrics
 - 单卡
 - 只读取少量数据即可(100张图片)
 - 标准的调整学习率的方法