针对脑瘤患者存活率预测

test_set

Model	loss	Bi_accuracy	AUC	F1	mcc	sensitivity	speficity
CNN	0.7261	0.6730	0.7362	0.4694	0.4452	0.4800	1.0000
ViT	1.0615	0.7547	0.7967	0.6486	0.5725	0.3067	**1.0000**
Swin	**0.5162**	**0.8302**	**0.8562**	**0.8000**	**0.6679**	**0.7200**	0.9286

首先基于脑癌患者病理切片数据集可以看出，Swin-transformer在所有性能上都远超CNN。尤其是最重要的MCC指标提升了近***50%***

肾脏移植存活率预测

i_score validation_set

Model	loss	AUC	balanced	mcc	sensitivity	speficity
CNN	0.6312	**0.9101**	0.7564	0.5788	0.5556	0.9573
ViT	4.2849	0.8644	0.7051	0.5090	0.4444	**0.9658**
Swin	**0.4442**	0.9044	**0.8234**	**0.6141**	**0.7407**	0.9060

i_score test_set

Model	loss	AUC	balanced	mcc	sensitivity	speficity
CNN	0.6117	**0.9498**	**0.8094**	**0.4748**	**0.6939**	0.9250
ViT	2.2149	0.9251	0.6410	0.2848	0.3265	**0.9556**
Swin	**0.3506**	0.9427	0.6770	0.3249	0.4082	0.9458

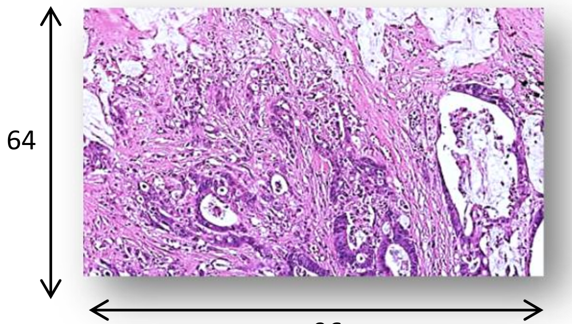
首先需要说明的是肾脏移植病理切片数据集由于来自不同医院的技术流程，所以图像风格尤其是色彩有明显差异，并且数量比例失衡即所谓的失衡数据集。因此Swin在验证集上有着明显提升，但是在测试集上表现欠佳，尤其体现为sensitivity过低。因此我编写了基于one-hot label的focal loss来平衡。

肾脏移植存活率预测-分类集 点损失函数纠正

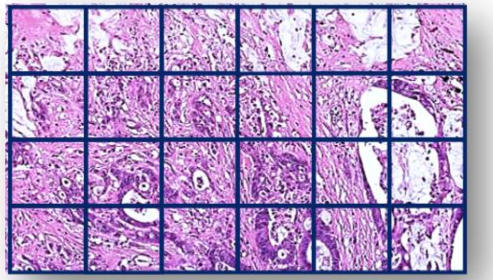
i_score test_set

Model	loss	AUC	balanced	mcc	sensitivity	speficity
Categorical	0.3506	**0.9427**	0.6770	**0.3249**	0.4082	**0.9458**
Focal	0.0272	0.9222	**0.7334**	0.3153	**0.5981**	0.8750

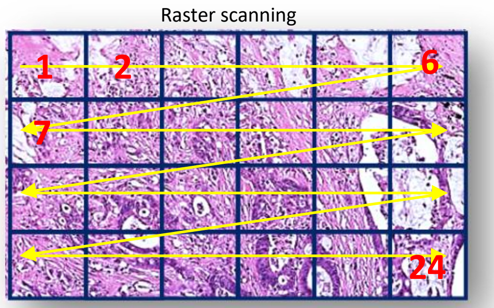
ViT基于WSIs工作流程讲解



input image
(64x96 pixels)



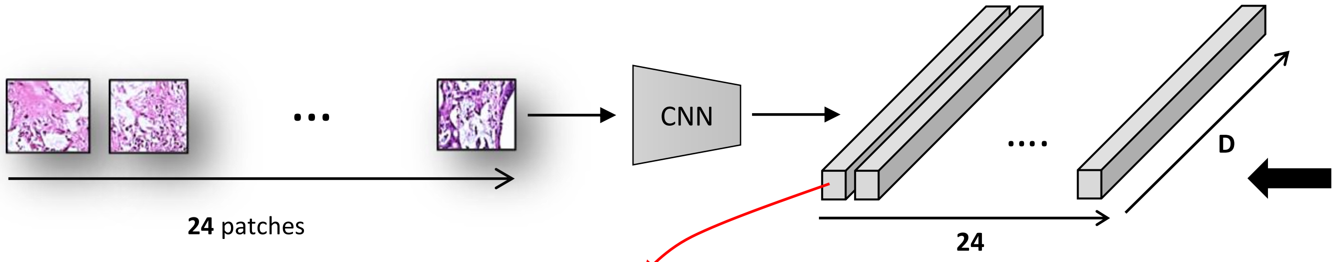
First step: Patching
Let's consider patch size is (16x16), so we will have patch grid with the size of (4x6)



Second step: Reshaping
Here we convert 2d patch grid into a patch sequence. We can use raster scanning, or any other types of scanning to rearrange patches from 2d into 1d sequences.
Important NOTE: the scanning method must be the same throughout dataset, and for example first tile should point to the same location on all input images. (if images have different sizes, it would be challenging to handle this)

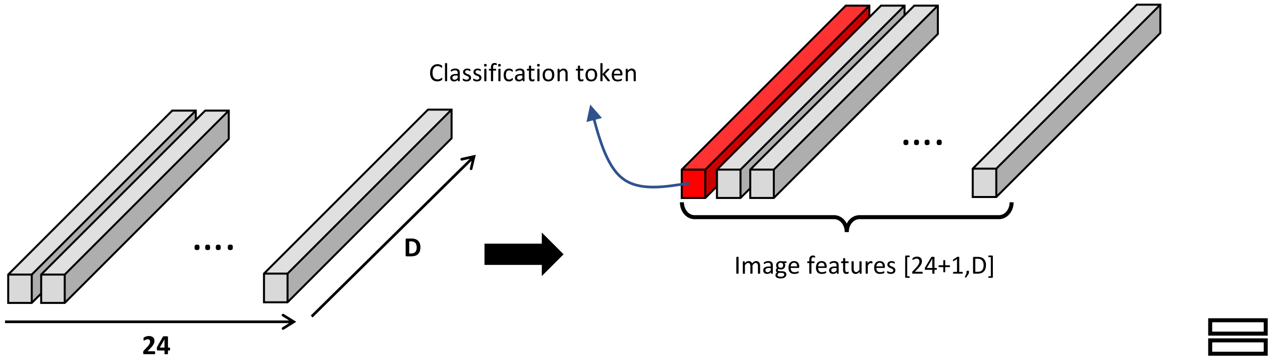
Patch sequence with the length of 24

Third step: Feature encoding
In this steps some high level features will be extracted from the patches. We can simply reshape patches themselves into 1d vector, but this might not be the efficient way to extract features. Instead of that, we can use a pre-trained CNN to do that.



NOTE: we call each of these feature vectors a token

如何在embedding中标准不同输入尺寸



Fourth step: Positional encoding
In this step, we add a trainable classification token to the beginning of the sequence. It’s a initialized randomly but would be trained during model training. Then we add positional encoded features to the tokens.