Generated on 2025-05-28

ParaN3xus's Blog 2024

Archive of Blog posts in 2024

目录

2024-12-31

UniMERNet 源码阅读

本文记录了对 UniMERNet 源码的阅读和理解, 包括类层次结构和四点主要改进.

UniMerNet 是一个针对数学公式的 TrOCR 模型. 基本上, 他是一个 Donut 的变体, 包含一个修改过的 Swin Encoder 和一个修改过的 BART Decoder.

由于他们的官方代码大量从 transformers 库中复制, 所以非常混乱, 嵌套了数不清层的类, 所以专门写一下 Blog 记录我一中午的阅读成果.

1.1 类层次

```
model: unimernet.UniMERModel
    tokenizer: encoder_decoder.DonutTokenizer
    model: encoder_decoder.DonutEncoderDecoder
        model: encoder_decoder.CustomVisionEncoderDecoderModel
            encoder: encoder_decoder.VariableUnimerNetModel (SwinModel - layernorm +
*embeddings)
                num_layers: int
                num_features: int
                embeddings: encoder_decoder.VariableUnimerNetEmbeddings
(SwinEmbeddings + *patch_embeddings - interpolate_pos_encoding)
                    patch_embeddings:
encoder_decoder.VariableUnimerNetPatchEmbeddings (SwinPatchEmbeddings + StemLayer)
                        projection: encoder_decoder.StemLayer (FGE)

                encoder: UnimerNetEncoder (SwinEncoder + *UnimerNetStage)
                    layers: [UnimerNetStage (SwinStage + ConvEnhance)]
                        blocks: [UnimerNetLayer (SwinLayer + ConvEnhance +
shift_size=0)]
                            shift_size: 0 (RSW)
                            layernorm_before: LayerNorm
                            ce: [ConvEnhance] (CE)
                            attention: SwinAttention
                            drop_path: SwinDropPath
                            layernorm_after: LayerNorm
                            intermediate: SwinIntermediate
                            output: SwinOutput

                pooler: AdaptiveAvgPool1d

            decoder: encoder_decoder.CustomMBartForCausalLM
                model.decoder: modeling_unimernet_decoder.MBartDecoder (or
CustomMBartDecoder) (BardDecoder - spda + squeeze_attn + layernorm + count(todo,
currently none))
                    embed_tokens: BartScaledWordEmbedding
                    embed_positions: BartLearnedPositionalEmbedding
                    layers: [MBartDecoderLayer]
                        *_attn: MBartSqueezeAttention / MBartFlashAttention2 (SA)
                    layernorm_embedding: LayerNorm
```

```
                    layer_norm: LayerNorm

processor: unimernet.processors.formula_processor.FormulaImageEvalProcessor
```

上述层次图基本展示了重要功能模块的组成, 并标注了论文中宣称的 FGE, RSW, CE, SA 对应在源码中的具体位置.

1.2 四点改进

1.2.1 Fine-Grained Embedding(FGE)

UniMerNet 把 Swin Encoder 中 “把图片分为不重叠的 Patch + 线性映射”(`PatchEmbeddings` 中的 `projection`)的操作更换为两次卷积:

```python
class StemLayer(nn.Module):
    """
    Stem layer of InternImage
    Args:
        in_chans (int): number of input channels
        out_chans (int): number of output channels
        act_layer (str): activation layer
        norm_layer (str): normalization layer
    """

    def __init__(self,
                 in_chans=3,
                 out_chans=96,
                 act_layer=nn.GELU,
                 norm_layer='BN'):
        super().__init__()
        self.conv1 = nn.Conv2d(in_chans,
                               out_chans // 2,
                               kernel_size=3,
                               stride=2,
                               padding=1)
        self.norm1 = build_norm_layer(out_chans // 2, norm_layer)

        self.act = act_layer()
        self.conv2 = nn.Conv2d(out_chans // 2,
                               out_chans,
                               kernel_size=3,
                               stride=2,
                               padding=1)

    def forward(self, x):
        x = self.conv1(x)
        x = self.norm1(x)
        x = self.act(x)
        x = self.conv2(x)
        return x
```

把 patch 换成卷积已经是一个很常见的魔改了, 据说好处很多, 能加快收敛, 提高表现等等, 一个讨论见 Early Convolutions Help Transformers See Better.

1.2.2 Convolutional Enhancement(CE)

UniMerNet 认为 Transformer 能较好地捕捉全局信息, 但是对于数学公式识别来说, 一些局部信息(小的上下标等)也很重要. 所以, 他们在每个 Swin Layer 的 Window Attention 和 MLP 层之前都加了一个 Kernel Size = 3*3, Stride = 1 的卷积, 也即 Convolutional Enhancement 模块:

```
class ConvEnhance(nn.Module):
    """
    Depth-wise convolution to get the positional information.
    """
    def __init__(self, config, dim, k=3):
        super(ConvEnhance, self).__init__()
        self.proj = nn.Conv2d(dim,
                              dim,
                              (k,k),
                              (1,1),
                              (k // 2,k // 2),
                              groups=dim)
        self.act_fn = ACT2FN[config.hidden_act]

    def forward(self, x, size: Tuple[int, int]):
        B, N, C = x.shape
        H, W = size
        assert N == H * W

        feat = x.transpose(1, 2).view(B, C, H, W)
        feat = self.proj(feat)
        feat = self.act_fn(feat)
        feat = feat.flatten(2).transpose(1, 2)

        x = x + feat
        return x
```

这里的激活函数选用的是 GELU.

1.2.3 Removal of Shift Window(RSW)

Swin 原版设计 Shift Window based Multi-Head Self-Attention(SW-MSA) 是想解决多个 Window 之间互相沟通的问题. 由于前面的魔改主要是加入了大量的卷积, 多个 Window 之间已经有了沟通, 或者说“模型的感受野已经很大了”, 所以这个模块也就没必要存在了, 删掉还能加速. 此外根据他们的实验, 删掉之后模型表现也会提升.

官方的实现没有直接删掉相关代码, 而是把 `SwinLayer` 的 `shift_size` 参数设置为 `0` 来关掉这个步骤:

```
class UnimerNetStage(nn.Module):
    def __init__(self, config, dim, input_resolution, depth, num_heads, drop_path,
downsample):
        super().__init__()
        self.config = config
        self.dim = dim
        self.blocks = nn.ModuleList(
            [
                UnimerNetLayer(
                    config=config,
                    dim=dim,
```

```
                    input_resolution=input_resolution,
                    num_heads=num_heads,
                    shift_size=0,
                )
                for i in range(depth)
            ]
        )

        = patch merging layer
        if downsample is not None:
            self.downsample = downsample(input_resolution, dim=dim,
norm_layer=nn.LayerNorm)
        else:
            self.downsample = None

        self.pointing = False
```

1.2.4 Squeeze Attention(SA)

这是一个用于提速的改进. 原本的 BART Attention 的 `q` 和 `k` 是和 `embed_dim` 一样大的. 这可能有点多余了, 所以 UniMerNet 中直接把这个维度砍半, 实验发现性能损失很小, 但是推理速度快了不少. 代码大部分都是照搬 BART Attention, 只是在相关的地方修改了 shape 而已, 这里就不贴了.

1.3 干净实现

Repo.

主要是删除了大量复制的代码, 能继承 transformers 的就继承. 此外, 还把原版自己造的接口换成了 transformers 类似的接口, 包括 `VisionEncoderDecoder` 和 `Processor` 等.

2024-12-30

我与 Typst 的 2024

回顾 2024 这一年我在 Typst 社区的成长历程, 从普通用户到社区贡献者的转变.

很早之前就读过了 @mgt 大佬的 我与 typst 的 2023, 觉得很酷, 于是决定自己也写一个. 不过当时是年初, 那时候我还只是一个普普通通的 Typst user, 没什么能说的出来的贡献. 时间一晃就到了现在, 我也总算是做了一点微薄的贡献, 可以拿出来说一说了.

2.1 2023-9 到 2024-2: 故事的开始

我大概就是这个时候知道 Typst 的. 那时候在几位前辈的努力下, Typst 的编写体验已经非常完美, 尤其是实时预览直接把写 Typst 的幸福度提高到把 LaTeX 甩到没影.

当时主要吸引我的就是他的数学公式语法, 不用一直 \ 和 {} 真的爽太多, 正好我需要一个写数学作业的软件, 于是就这样用上了 Typst.

说来也算是缘分吧, 最开始我只是把 Typst 拿来写数学作业, 又恰好是英文写, 所以我直接拿官网的默认模板就开写了. 也正是因为这样, 初期的各种中文问题我一点都没碰上, 因为调整排版也不多, 所以虽然不怎么懂其他方面的 Typst 用法, 也用了几个月.

2.2 2024-3 到 2024-6: 一个 Typst 信徒的诞生

这段时间我突然有了写实验报告的需求, 因为这个实验报告对格式的要求并不严格, 所以我就简单手搓了一个模板来写.

实验报告怎么也沾点学术排版的边了, 这下总算是把 Typst 里面各种东西怎么用, 什么地方有什么坑都搞了一遍. 虽然说官方文档还是没完整读过, 小蓝书也没完整看过, 但是总算是能用 Typst 正常排版了.

这段时间我逐渐变成 Typst 狂热粉丝, 在学校里见人就推销 Typst, 还把 Typst Introduction in One Page 改了一份印在了衣服上(不知道会不会侵权), 基本上天天穿着. 有次看到教室里的同学居然在安装 LaTeX, 马上向他安利, 并且亲手帮他把 LaTeX 卸载(现在想来真是太逆天了). 甚至英语课的 Poster Speech 都在安利 Typst.

我的狂热确实是有用的, 我恐怕是我小圈子里第一个开始用 Typst 的, 经过我这一年的软磨硬泡, 现在至少有五个人拜入 Typst 神教了.

2.3 2024-7 到 2024-8: Typst 公式 OCR, Typress

从这里开始总算对我狂热的社区有了一点贡献.

其实做 Typst 公式 OCR 的想法很早就有了. 我记得是 4 月, 本来打算做一个 Typst 的公式 OCR 当课程作业交, 但是当时的想法太简单了, 随手拿了个 pandoc 转了点 LaTeX 公式就开始训练. 最开始我只是想先试试水, 就拿了 GitHub 上星星比较多的 LaTeX-OCR 没改动就开始嗯训了, 但是最后结果毫无疑问是一大坨, 也就暂时放弃了这个想法.

但是我一直很想做, 于是在放暑假之后和 @sjfhsjfh 又重新开始了这个项目.

其实公式 OCR 已经没什么秘密了. HuggingFace 上 TrOCR 嗯造出效果很好的 LaTeX OCR 模型有三四个. 但是他们有数据, 而 Typst 没有.

所以我和 sjfhsjfh 兵分两路, 想先解决数据的问题.

2.3.1 eq_query_rec

我们首先想到的就是从真人手里拿数据. 但是当时大家普遍都有自己定义的小 snippets, 或者在使用各种第三方包(尤其是 physica), 我们不想让数据包含这些内容, 但是直接筛掉的话绝大多数数据都没有了, 所以写了这样的一个工具来提取公式.

得益于 Typst 的强大功能, sjfhsjfh 写了 eq_query_rec, 他可以编译 Typst 文档, 然后从里面查找出所有公式, 再根据公式编译好的内容重组出标准化后的公式(去除了第三方包, 用户自己的函数, 并同时完成公式形式的标准化).

这在初期帮了我们大忙. 我们最开始仅有的 8K 条数据全都是小圈子里几个 Typst user 用这个工具提取的.

2.3.2 tex2typ

Typst user 的数量太有限了, 而且大群(指 Typst 非官方中文交流群)的群友也不是很乐意搭理我的数据乞讨, 而另一边, LaTeX OCR 模型的训练数据量有 M 级别, 所以要解决数据问题最后还是要靠转的.

其实转公式的工具已经有很多了, 就我而言, 我至少知道四个工具能达成这个任务. 但是 pandoc 效果太烂, mitex 比较专注视觉效果一致, 公式本身不像人写的, 还包含很多他自己定义的函数, 其他的效果也很一般. 我真正想要的是转出来的公式不仅视觉效果要一样, 还得必须“像人写的”, 尤其是“像 Typst 用户”写的.

所以最后我借用 KaTeX 的 parser 写了这个名字很豪的 tex2typ, 但是当时写的时候太急, 回过来味的时候发现已经写成史山了. 不过好在我们并不要求所有公式都能转, 毕竟是数据集, 大部分还在就 OK, 最终我们用这种方法获得了(疑似是)首个 100K 级别和 M 级别的 Typst 公式数据集.

2.3.3 Typress

我一直记得有一个群友抱怨 “没人想要抄写这一串东西(很长的公式)”, 所以当第一版效果还行的模型训练出来之后(大概是七月中), 我就去群里报喜了. 没想到最激动的是群主, 不知道我有没有辜负他的期待(.

Typress 发出来是七月底的事情了, 当时只配了一个 GPT 生成的简陋网页前端, 不过好在能用, 效果也还行.

在那之后就是慢慢改善模型对 `mat`, `cases` 之类长难句的识别以及写更好的前端, 修代码里的小 bug 了, 不过现在我手里还有一版对手写支持更好(比原本模型好太多)的权重没有发布, 或许等到寒假再优化一下再发?

2.4 2024-9 到 2024-10: tinymist-nightly

那时候的 Typst, 虽然开发还在积极推进, 但是已经有很久没发版了, `v0.11.1` 是五月中, 到九月已经足足有四个月, 这可是三分之一年.

当时群里有很多 Frequent asked questions, 比如 breakable equations, performance, layout issue 等等, 都是已经修了但还没发版的状态.

很多普通用户不会自己编译 Typst, 所以我最开始就想要不维护个 typst-nightly 的仓库, 写个定时的 action 自动发版. 但是我转念一想, 这样的帮助并不大: 我们使用的时候都是 tinymist 负责编译, 要是用到了新特性可能干脆就编译错误, 然后 preview 也不能正常更新, 或者预览效果和预期并不一致. 总不能看着旧版的 preview 写, 最后用 nightly 的 typst 来编译吧, 这完全没意义, 所以我就想干脆把 tinymist 适配到最新 Typst 算了.

其实我一开始以为就是改个依赖版本就好了, 没想到 Typst 在这几个月里面做了这么多 API 变动. 再加上这一套工具全都是群友科技, tinymist 依赖的 typst.ts, typstyle 也要一并修改, 所以我干脆就全都给适配到最新 Typst 了.

我把这个发到群里的时候, tinymist 作者 @纸夜 大神提出可以做一个 `typst-shim` 包, 来减少 tinymist-nightly 和主线的摩擦, 然后我一个 Rust 小白就这样混上了 tinymist contributor. 后来纸夜大神又提出可以把一部分版本号给 tinymist-nightly, 然后我这个分支就成功转正, 进入了官方 tinymist 的仓库.

再后来, 我的名字也被加到了 `MAINTAINERS.typ` 里面, 虽然好久没干活了(心虚), 预计寒假再开始工作.

2.5 2024-11: Typress 的公式检测

当时手头各种工作告一段落, 就抽了点时间借用其他公式 OCR 软件的公式检测模型给 Typress 用. 同时也在前端支持了手动调整剪裁公式(虽然大部分代码都是 LLM 代劳). 现在使用 Typress 终于不需要截图时手动框选公式了.

2.6 2024-11 到 ∞: 我还在继续 Typst

其实这些工作是穿插在上面的时间线中的.

自从 Typst 有了官方论坛, 我就开始在上面疯狂答题, 目前也是成功收获 Guidance Counsellor 称号(贡献 Solutions 23 个). 我想大概论坛的活跃用户都认识我这个 ID 了.

另外, 还用 Typst 搓了一堆小玩意, 但是都是小群内部偷偷用, 没有公开出来, 可能寒假会看看, 比如红头文件模板, p5 预告信之类的.

我的日常学习已经和 Typst 完全融合了, 这学期我所有有必要做笔记的科目都用 Typst 写了笔记(虽然大部分抄书). 另外, 还充分发挥了 https://typst.app 的多人编辑优势, 把 Typst 作为打比赛时团队管理的有效工具. 我多个科目的作业也都用 Typst 撰写.

希望明年的 Typst 和我继续变强!

2024-08-24

Wolfram Mathematica 14.1 Key Generator

使用 Wolfram Mathematica 14.1 的密钥生成器, 支持多种版本的 Mathematica 和 System Modeler.

最近 Wolfram 发布了新的 Mathematica 14.1 版本, 打破了之前不需要改注册机就能一直用的局面, 不过所幸毛子又发了新的[^1], 目测好像就是更新了以下 magic 值, 算法并无大的改动.

我费了一番力气才找到, 为了方便他人搬运到这里, 原版链接.

使用方法和以前一样, 下方的激活码处保留了一个默认值, 如果手动删除留空则会随机生成.

2024-08-17

具体数学第一章课后习题详解

具体数学第一章课后习题的详细解答

最近在拜读高老头的大作《具体数学》, 并力所能及地完成习题. 习题的具体步骤将会以 Blog 文章的形式发出来, 就像这篇. 考虑到考试题可能涉及多个章节的内容, 第一遍通读我会先行跳过, 之后再来补上.

由于这一系列 Blog 文章都使用 Typst 排版, 所以网页上的显示效果可能会略差, 请多包涵.

6. 热身题

a.

“标号从 1 直到 $n-1$ 的马都有同样的颜色” 并不能推出 “标号从 2 直到 n 的马都有同样的颜色”. 实际上, 这直接假定了我们要证明的结论.

b.

考虑到要把整个塔移动到 B 柱, 首先要把最底下的盘移动到 B 柱. 移动最底下的盘时, 必须先移动到中间的柱, 然后将 B 柱上的盘移动回 A 柱, 再将最底下的盘移动到 B 柱. 然后再将剩余的盘移动到 B 柱.

设移动 x 个盘的最短步数为 $H(x)$, 那么我们有:

$$H(x) = H(x-1) + 1 + H(x-1) + 1 + H(x-1) = 3H(x-1) + 2 \qquad (x > 1)$$
$$H(1) = 2$$

这是我们得到的递归式, 为了得到一个封闭形式, 我们可以:

$$\begin{aligned} H(x) + 1 &= 3H(x-1) + 3 \\ &= 3(H(x-1) + 1) \\ &\Rightarrow H(x) + 1 = 3^x \\ &\Rightarrow H(x) = 3^x - 1 \end{aligned}$$

与此同时, 我们也得到了具体的移动策略:

```
function Hanoi(A, M, B, n)
    ▷ Move top n plates from A to C
    if n = 1 then
        Move(A, M)
        Move(M, B)
        return
    end
    Hanoi(A, M, B, n − 1)
    Move(A, M)
    Hanoi(B, M, A, n − 1)
    Move(M, B)
```

```
    | Hanoi(A, M, B, n − 1)
end
```

c.

这个题目的描述有些模糊, 我认为应当删除 “在 3 根桩柱上都” 几个字, 因为这样描述隐含了一种 “x 种排列在三根桩柱上各出现一次, 共 $3x$ 次” 的意味, 这样会把研究对象转为 “一根桩柱上的排列”, 然而实际上作者要求我们思考的是 n 个圆盘作为整体, 在三个桩柱上的排列情况.

容易发现 n 个圆盘的排列方式共有 3^n 种, 除去初始状态之外, 剩余的 $3^n - 1$ 种排列恰好与上述的最短步长相等. 于是我们只需要说明上述每一步的末状态均不相同. 而这是显而易见的, 因为如果相同, 那么上述步骤就不是最短步骤.

d.

如果以 n 个圆盘的某种叠放方式为结束状态, 那么

- 当 $n = 1$ 时, 显然最多只需要 1 步就能达成, $1 \leq 2^1 - 1$.
- 设当 $n = k$ 时, 最多需要 $2^k - 1$ 步达成, 那么当 $n = k + 1$ 时,
  - 若要求最大的圆盘在左侧柱桩, 则同样需要至多 $2^k - 1 < 2^{k+1} - 1$ 步达成.
  - 否则, 需要先把其他圆盘移动到剩余的一个空闲柱桩, 再将最大的圆盘移动到要求位置, 再将剩余圆盘移动到要求位置, 需要至多 $2^k - 1 + 1 + 2^k - 1 = 2^{k+1} - 1$ 步达成.

综上所述, 以 n 个圆盘的某种叠放方式为结束状态, 至多需要 $2^n - 1$ 次移动. 若以 n 个圆盘的某种叠放方式为开始状态, 则将 “以 n 个圆盘的某种叠放方式为结束状态” 时得出的步骤倒放即可, 同样至多需要 $2^n - 1$ 次移动.

故不存在 n 个圆盘在 3 根桩柱上的某种开始叠放或结束叠放, 使得按照卢卡斯原来的规则, 需要多于 $2^n - 1$ 次的移动.

e.

根据平面图的 Euler 定理, 对于任意连通图, $V - E + F = 2$, 其中 $E = 2V$, 则

$$\begin{aligned} V - 2V + F &= 2 \\ F &= V + 2 \end{aligned}$$

也即, **面数** $=$ **顶点数** $+ 2$.

而两个圆之间至多有 2 个交点, 对于四个圆, 则至多有 $2 + 4 + 6 = 12$ 个交点, 故至多把平面分成 14 个面. 因此并不能描述.

f.

至少有 3 条直线相交才能确定一个有界的区域, 所以对于 n 条两两相交的直线, 确定的有界区域数 F_n:

$$F_n = F_{n-1} + n - 2 \qquad (n > 1) F_1 = 0$$

故有

$$\begin{aligned} F_n &= F_{n-1} + n - 2 \\ &= F_{n-2} + n - 3 + n - 2 \\ &= F_{n-3} + n - 4 + n - 3 + n - 2 \\ &= F_1 + 0 + 1 + 2 + \cdots + n - 2 \\ &= \frac{(n-1)(n-2)}{2} \end{aligned}$$

g.

$$H(1) = J(2) - J(1) = 1 - 1 = 0$$

第一项不成立.

7. 作业题

a.

多列几项就会发现数列 $\{Q_n\}$ 是以 5 为周期循环的, 具体来说是

$n \bmod 5$	0	1	2	3	4
Q_n	α	β	$\frac{1+\beta}{\alpha}$	$\frac{1+\alpha+\beta}{\alpha\beta}$	$\frac{1+\alpha}{\beta}$

b.

4.2.2.1.

令 $x_n = (x_1 + x_2 + \cdots + x_{n-1})$, 则根据 $P(n)$, 有

$$\begin{aligned} x_1 x_2 \cdots x_n &= \frac{x_1 x_2 \cdots x_{n-1}(x_1 + x_2 + \cdots + x_{n-1})}{n-1} \\ &\le \left(\frac{x_1 + x_2 + \cdots + x_{n-1} + \frac{x_1 + x_2 + \cdots + x_{n-1}}{n-1}}{n} \right)^n \\ &\le \left(\frac{x_1 + x_2 + \cdots + x_{n-1}}{n-1} \right)^n \\ x_1 x_2 \cdots x_{n-1} &\le \left(\frac{x_1 + x_2 + \cdots + x_{n-1}}{n-1} \right)^{n-1} \end{aligned}$$

$P(n-1)$ 成立, 故 $P(n)$ 蕴涵 $P(n-1)$.

4.2.2.2.

令 $x_1 = y_1 + y_2 + \cdots + y_n$, $x_2 = y_{n+1} + y_{n+2} + \cdots + y_{2n}$, 则由 $P(2)$, 有

$$y_1 y_2 \cdots y_n y_{n+1} y_{n+2} \cdots y_{2n} \le \left(\frac{y_1 y_2 \cdots y_n + y_{n+1} y_{n+2} \cdots y_{2n}}{2} \right)^2$$

又根据 $P(n)$, 有

$$\begin{aligned}\text{LHS} &\le \left(\frac{\left(\frac{y_1+y_2+\cdots+y_n}{n}\right)^n+\left(\frac{y_{n+1}+y_{n+2}+\cdots+y_{2n}}{n}\right)^n}{2}\right)^2\\ &\le \left(\frac{\left(\frac{y_1+y_2+\cdots+y_{2n}}{n}\right)^n}{2}\right)^2\\ &\le \frac{(y_1+y_2+\cdots+y_{2n})^{2n}}{4n^n}\\ &\le \left(\frac{y_1+y_2+\cdots+y_{2n}}{2n}\right)^{2n}\end{aligned}$$

$P(2n)$ 成立, 故 $P(n)$ 和 $P(2)$ 蕴涵 $P(2n)$.

4.2.2.3.

$P(2)$ 成立, 根据 小节 4.2.2.2. 的结论, $P(2^n)$ 对于任意大的 n 也成立, 再根据 小节 4.2.2.1. 的结论, 对于任意大的 n, 都有 $P(x)$ 在 $x \le 2^n$ 时成立.

于是 $P(n)$ 对任意 n 成立.

c.

实际上 R_n 相当于将 n 个圆盘顺时针移动一个桩柱的最少移动次数, Q_n 则是将 n 个圆盘逆时针移动一个桩柱的最少移动次数.

Q_n 和 R_n 在 $n = 0$ 时的值显然是 0, 因为这不需要任何移动.

若 $n \neq 0$, 要把 n 个圆盘顺时针移动一个桩柱, 则需要先把最大的圆盘顺时针移动一个桩柱, 这时 B 柱必须是空的, 因此首先要把其余 $n-1$ 个圆盘逆时针移动一个桩柱, 再将最大的圆盘顺时针移动一个桩柱, 然后再把其余圆盘逆时针移动一个桩柱, 这时其余的圆盘就在 B 柱上了. 也即 $Q_n = 2R_{n-1} + 1$. 所有步骤如下表格

步骤	步数	A	B	剩余的一个盘
0		$n, n-1, \cdots, 1$		
1	R_{n-1}	n		$n-1, n-2, \cdots, 1$
2	1		n	$n-1, n-2, \cdots, 1$
3	R_{n-1}		$n, n-1, \cdots, 1$	

若要把 n 个圆盘逆时针移动一个桩柱, 步骤与上一种情况类似. 首先需要把最大的圆盘逆时针移动一个桩柱, 实际操作时则是顺时针移动两个桩柱, 这就需要 B 桩柱是空的. 所以首先我们要把其余 $n-1$ 个圆盘逆时针移动一个桩柱, 再将最大的圆盘顺时针移动一个桩柱, 接下来我们需要再把最大的圆盘顺时针移动一个桩柱, 这需要第三个桩柱是空的, 所以我们需要把其余 $n-1$ 个桩柱先顺时针移动一个桩柱, 再把最大的圆盘顺时针移动一个桩柱, 最后我们需要把其余 $n-1$ 个圆盘移动到 B 桩柱, 所以最终的步骤数为 $R_n = R_{n-1} + 1 + Q_{n-1} + 1 + R_{n-1} = Q_n + Q_{n-1} + 1$. 所有步骤如下表格

步骤	步数	A	B	剩余的一个盘
0		$n, n-1, \cdots, 1$		
1	R_{n-1}	n		$n-1, n-2, \cdots, 1$
2	1		n	$n-1, n-2, \cdots, 1$

3	Q_{n-1}	$n-1, n-2, \cdots, 1$	n	
4	1	$n-1, n-2, \cdots, 1$		n
5	R_{n-1}			$n, n-1, \cdots, 1$

d.

4.2.4.1.

考虑到要把整个塔移动到 B 柱, 首先要把最底下的两个盘移动到 B 柱. 移动最底下的盘前, 必须先把其余 $n-1$ 个大小的盘移动到另一个柱子, 然后将 A 柱上的两个最大的盘移动到 B 柱, 然后再将剩余的盘移动到 B 柱.

设移动 n 个尺寸的圆盘的最小步数为 $H(n)$, 则有

$$\begin{aligned} H(1) &= 2 \\ H(n) &= 2H(n-1) + 2 \qquad n > 1 \end{aligned}$$

这是我们得到的递归式, 为了得到一个封闭形式, 我们可以:

$$\begin{aligned} H(n) &= 2H(n-1) + 2 \\ H(n) + 2 &= 2H(n-1) + 4 \\ H(n) + 2 &= 2(H(n-1) + 2) \\ H(n) + 2 &= 2^{n+1} \\ H(n) &= 2^{n+1} - 2 \end{aligned}$$

4.2.4.2.

暂时跳过

e.

与 小节 4.2.4.1. 同理, 我们有

$$\begin{aligned} A(x) &= x \\ A(m_1, \cdots, m_n) &= 2A(m_1, \cdots, m_{n-1}) + m_n \qquad n > 1 \end{aligned}$$

这是我们得到的递归式, 为了得到一个封闭形式, 我们可以:

$$\begin{aligned} A(m_1, \cdots, m_n) &= 2A(m_1, \cdots, m_{n-1}) + m_n \\ &= 4A(m_1, \cdots, m_{n-2}) + 2m_{n-1} + m_n \\ &= 2^{n-1}m_1 + \cdots + 2m_{n-1} + m_n \end{aligned}$$

故 $A(m_1, \cdots, m_n) = \sum_{i=1}^{n} 2^{n-i} m_i$.

f.

我们可以构想一个更大的圆把所有 Z 包围, 然后根据平面图的 Euler 定理, 对于任意连通图, 有

$$V - E + F = 1$$

这里我们忽略大圆外面的一个面, 所以等式的右侧是 1.

关于 E 和 V, 我们可以根据几种不同的交点讨论求出. 经过观察, 交点的类型和对应的度有以下几种:

- Z 和外面大圆的交点: $2n$ 个, 度为 3

- Z 自身的两个折点: $2n$ 个, 度为 2
- 两个 Z 之间的交点: x 个, 度为 4 (为了让产生的面最多, 我们保证任意两个 Z 的三条线都是相交的)

于是我们有

$$2n + 2n + x - \frac{3 \cdot 2n + 2 \cdot 2n + 4x}{2} + F = 1$$
$$F = 1 + n + x$$

接下来我们只需要求出 x. 经过观察, 我们发现两个 Z 之间最多产生 9 个交点, 又根据两个 Z 的三条线都相交, 我们可以得知

$$x_1 = 0$$
$$x_n = x_{n-1} + 9(n-1) \qquad n > 1$$

我们把递归式展开

$$\begin{aligned} x_n &= x_{n-1} + 9(n-1) \\ &= x_{n-2} + 9(n-2) + 9(n-1) \\ &= x_{n-3} + 9(n-3) + 9(n-2) + 9(n-1) \\ &= x_1 + 9 + \cdots + 9(n-3) + 9(n-2) + 9(n-1) \\ &= \frac{9n(n-1)}{2} \end{aligned}$$

然后再代入原式

$$\begin{aligned} F &= 1 + n + x \\ &= 1 + n + \frac{9n(n-1)}{2} \\ &= \frac{2 - 7n + 9n^2}{2} \end{aligned}$$

g.

考虑到当我们添加一个新的平面时, 新增的某个多面体不会同时有两个面在这个新的平面上, 因此新增的多面体数实际上等于该平面被其他平面截出的区域数, 也即

$$P_1 = 2$$
$$P_n = P_{n-1} + L_{n-1}$$

h.

根据与正文中问题相同的讨论步骤, 我们有

$$\begin{cases} I(2n) = 2I(n) - 1 \\ I(2n+1) = 2I(n) + 1 \end{cases} \qquad \begin{cases} I(2) = 2 \\ I(3) = 1 \end{cases}$$

参考书中的方法, 我们可以设 $\beta_0 = -1, \beta_1 = 1$, 于是递推式可以化成

$$I(2n + j) = 2I(n) + \beta_j$$

参考书中的方法, 我们用二进制表示 n

$$n = (b_m b_{m-1} \cdots b_1 b_0)_2$$

则

$$\begin{aligned} I\Big((b_m b_{m-1}\cdots b_1 b_0)_2\Big) &= 2I\Big((b_m b_{m-1}\cdots b_1)_2\Big) + \beta_{b_0} \\ &= 4I\Big((b_m b_{m-1}\cdots b_2)_2\Big) + 2\beta_{b_1} + \beta_{b_0} \\ &= 2^{m-1} I\Big((b_m b_{m-1})_2\Big) + 2^{m-2} b_{m-2} + \cdots + 2\beta_{b_1} + \beta_{b_0} \\ &= \begin{cases} 2^{m-1} + \sum_{i=0}^{m-2} \beta_{b_i} & b_{m-1} = 1 \\ 2^m + \sum_{i=0}^{m-2} \beta_{b_i} & b_{m-1} = 0 \end{cases} \end{aligned}$$

i.

我们可以把通项的形式设为

$$g(n) = A\alpha + B\gamma + C\beta_0 + D\beta_1$$

考虑 $\gamma = 0$, 则本问题退化为正文中讨论过的问题, 可以得到

$$\begin{aligned} g\Big((b_m b_{m-1}\cdots b_1 b_0)_2\Big) &= \Big(\alpha\beta_{b_{m-1}}\beta_{b_{m-2}}\cdots\beta_{b_1}\beta_{b_0}\Big)_3 \\ &= 3^m\alpha + \sum_{i=0}^{m-1} \beta_{b_i} 3^i \\ &= 3^m\alpha + \sum_{i<m,\beta_i=0} 3^i b_0 + \sum_{i<m,b_i=1} 3^i\beta_1 \end{aligned}$$

令 $g(n) = n$, 则有

$$\begin{cases} 1 = \alpha \\ 2n = 3n + \gamma n + \beta_0 \\ 2n+1 = 3n + \gamma n + \beta_1 \end{cases}$$

由于上面的结果对于任意 n 都成立, 所以

$$\begin{cases} \alpha = 1 \\ \gamma = -1 \\ \beta_0 = 0 \\ \beta_1 = 1 \end{cases}$$

故

$$\begin{aligned} n &= A - B + D \\ B &= A + D - n \\ &= 3^m + \sum_{i<m,b_i=1} 3^i - n \end{aligned}$$

故

$$g(n) = 3^m\alpha + \left(3^m + \sum_{i<m,b_i=1} 3^i - n\right)\gamma + \sum_{i<m,\beta_i=0} 3^i b_0 + \sum_{i<m,b_i=1} 3^i\beta_1$$

8. 考试题

9. 附加题

10. 研究题

2024-08-01

NVIDIA + Hyprland + Arch 踩坑

使用 NVIDIA 显卡和 Hyprland 在 Arch Linux 上的经验分享

继 Debian + Gnome 用了没几天就换回 Windows 吃灰后, 又下定了一次决心日用纯 Linux 系统. 这一次我装了 Arch + Hyprland, 在 Arch 强大的生态和自定义能力的加持下终于也算用了一个月. 虽然系统还是有各种小问题无法解决, 但是稍微忍忍已经完全能用了. 所以这篇 Blog 就是记录我是怎么调教 NVIDIA + Hyprland + Arch 打到一个相对良好的状态的.

不过还有很多问题我没有解决, 也一同记录在下面, 就当作是一个 TODO List 了, 没事的时候我会来看看的.

5.1 NVIDIA 驱动开机崩溃

我因为有使用 CUDA 的需要, 所以安装了 NVIDIA 的闭源驱动, 然后就出现了这个问题.

主要表现为开机概率性的崩溃, 这个原因就是 NVIDIA 驱动和内核的兼容性不太好, 最终我在 `linux-lts + nvidia-dkms` 上实现了稳定.

本文写作时, 上述软件包的版本为:

- `linux-lts: 6.6.42-1`
- `nvidia-dkms: 555.58.02-1`

5.2 双屏幕双显卡的配置

本来我想核显一个屏幕, 独显一个屏幕, 但是无论怎么配置都没法正常用, 主要表现为要么两个屏幕都黑屏, 要么一个黑屏, 反正就是没法正常用.

最后终于在 Hyprland Wiki 上找到了明确的说明, 在 Windows 上如此自然的想法是不可能的, 我们必须关掉核显.[1]

5.3 双屏幕不同 DPI 下 XWayland 应用的缩放

这个要求主线目前做不了, 不过 Hyprland 社区已经有相关 PR[2]. 经过本人测试工作良好, 可按照 PR 内指引尝试.

5.4 QQNT 不能复制

这个问题是给 Electron 指定平台参数:

```
--enable-features=UseOzonePlatform --ozone-platform=wayland
```

引起的.

我曾经在各种社区或者 issue 里找到了一些用于修复复制粘贴问题的脚本, 但是这些脚本似乎都会破坏其他功能, 因此我没有最终使用.

我的建议是仍然使用 XWayland 模式, 虽然不能缩放, 但是至少能复制粘贴.

[1] Hyprland Wiki - Multi-Monitor with Hybrid Graphics
[2] hyprwm/Hyprland - xwayland: support HiDPI scale

5.5 Emoji Picker

我始终觉得输入 “haha” 来调用 Emoji 表情非常蠢而且尴尬, 所以还是选择找一个好用的 Emoji Picker.

经过一番寻找, 我发现这个还不错: [im-emoji-picker](https://github.com/GaZaTu/im-emoji-picker).

不过要让他在 Hyprland 下良好的工作, 还需要设置一些简单的 WindowRules, 这里是我的配置:

```
windowrulev2 = nofocus,initialTitle:^(im-emoji-picker)$
```

5.6 RIME 配置

虽然输入法本身和 Wayland 之类的关系不大, 但是作为我调整系统过程的一个重要部分, 我还是选择写在这里.

RIME 的配置用的是 “预构建配置 + 用户自定义 Patch” 的形式, 但是他们的文档讲的不是很清楚, 我花了很长时间才搞明白要怎么用.

我使用的预构建配置是: [rime-ice](https://github.com/iDvel/rime-ice).

我主要是取消了中英文切换的 Shift 快捷键, 然后修改了默认标点为半角, 并设置页面选项数为 9, 并且默认不显示那些我用不到的配置. 具体需要进行如下修改:

`~/.local/share/fcitx5/rime/default.custom.yaml`

```
patch:
  "menu/page_size": 9
  schema_list:
    - schema: luna_pinyin_simp

  "ascii_composer/switch_key/Shift_L": none
```

`~/.local/share/fcitx5/rime/luna_pinyin_simp.custom.yaml`

```
patch:
  "menu/page_size": 9
  "switches/@3/reset": 1

  "ascii_composer/switch_key/Shift_L": none
```

5.7 小键盘有时失灵

具体表现为小键盘灯亮着但是实际上不能用, 需要按一下关闭再按一下打开, 或者(有时)切换输入法再切回来.

因为后面这个解决方式, 我猜测这个和输入法有关, 但是目前还没得到验证, 也没找到靠谱的解决方式.

5.8 截图不显示鼠标失效 + 选择光标会被截上

我目前用 `grimblast` + `screenshot.sh`[3] 进行截图.

前一个问题我没什么头绪, 这似乎和 `screenshot.sh` 冻结屏幕的方式有关, 也有可能和 `hyprpicker` 有关. 我不太确定.

[3] screenshot.sh

后一个问题似乎与用于 freeze 屏幕的 `hyprpicker` 有关[4], 但是还没有修复. Github 的相关 issue[5] 有一些 workround, 但是对我没用.

5.9 截图后有时 `hyprpicker` 不能正常退出导致屏幕无法操作

这个比较容易修复, 但是我还没有实践, 主要是因为出现概率比较小 + 修复简单.

遇到这种情况切到其他 tty 然后手动 `kill` 掉 `hyprpicker` 进程即可.

5.10 Special Workspace 中 XWayland 应用遮挡输入法

这个是一个尚未修复 Bug, 在 Hyprland 仓库有相关 issue.

不过就作者态度来看, 他大概不太会想修这个 Bug 了.

5.11 float 窗口关闭时 focus 不会自动转移到鼠标所在的位置

~~我觉得这是个很典型的 Bug, 但是似乎 Hyprland 到现在也没有修复? 我也没有找到相关 issue.~~

我创建了一个 PR 用于添加一个选项指定窗口关闭时, focus 转移的行为. 现在只需要指定 `input::focus_on_close = 1` 就能实现想要的效果.

5.12 Chromium 首屏有概率渲染错乱

在 `~/.config/chromium-flags.conf` 文件中指定

```
--enable-wayland-ime
--ozone-platform=wayland
--use-angle=vulkan
```

但是这样做会导致 B 站不能播放视频, 目前还没有找到解决方案.

[4] hyprwm/hyprpicker - Add a flag to hide the cursor in hyprpicker
[5] hyprwm/contrib - Area selection cursor is shown in final screenshot

2024-07-05

已知模型结构的 ONNX 模型转 PyTorch 模型

通过直接找出 ONNX 模型和 PyTorch 模型的权重之间的 Map, 实现 ONNX 模型到 PyTorch 模型的转换.

6.1 缘由

我们已经有很多 ONNX 模型转 PyTorch 模型的工具了, 但是根据我的个人体验, 模型稍微复杂一点, 这些工具就寄了.

在我的一个任务中, 我有一个已知结构, 实现完全相同, 但是比较复杂的 ONNX 模型, 出于一些原因, 我需要把它转成对应这个实现的 Torch 模型, 这时现有的工具并不能很好的完成任务–这些模型都不能利用现有的模型实现, 而且应付不了复杂的模型.

6.2 过程

6.2.1 想法

由于我们的目的是逆向一个结构已知, 而且已经获得其实现的模型, 那我们自然可以用一些通用工具用不了的思路. 比如说, 我可以大胆猜测 ONNX 模型生成时使用了一种比较良好的方式, 使得这个转换过程是稳定, 一致的.

6.2.2 观察

经过对要转换的 ONNX 模型和 Torch 模型(随机初始化出来的) 的观察, 我发现:

- 两个模型的权重数量相近, 具体来说, 是 `Encoder` 部分差 2, 而 `Decoder` 部分相等, 而 `Encoder` 部分差的那两个是和池化相关的, 应该是原模型没有开启.
- ONNX 中的权重大多数都有可读的名字, 但是少数的名字变成了如 `onnx::MatMul_1234` 这种不可读的形式.

这里比较麻烦的就是这些名字丢失的权重, 不过因为权重的数量是相近的, 我还是猜测这些权重之间是一一对应的.

另外由于部分权重的名字仍然可见, 我关于 ONNX 模型生成方式的猜测得到了印证, 在我的例子中, 具体来说, 就是

```
optimum-cli export onnx --task image-to-text --model torch_model onnx_model
```

这个工具还是比较友好的, 他转出来的 ONNX 都是一致的, 这就方便了我们更直接地得知导出过程中到底发生了什么.

6.2.3 实验

我首先设法构造了这样的一个模型, 它的权重是以 0.1 递增的数字, 然后将其导出为 ONNX.

这时我发现 ONNX 模型中名称不可读的权重果然都与 PyTorch 中的某些权重对应, 前面提到的差 2 的权重无关紧要.

但是有一个问题是, 这些名字丢失的权重, 虽然内部的值(全都是 0.x)对得上 PyTorch 模型中的某个权重, 但是有一部分的 shape 却对不上, 具体来说, 是一些矩阵的大小转置了.

为了确认大小转置了是不是里面的值也确实转置了, 我又构造了另一个模型, 这个模型中的权重都是不对称的, 第一行全都是某个值, 其余行全都是另一个值, 这两个值递增且不相等.

这次的结果确认了我的想法, 那些名字丢失的矩阵就是被转置了. 关于为什么要转置, 我想可能是因为某种计算过程的改变:

$$AB = \left(B^T A^T\right)^T$$

总之我们找到了 ONNX 和 PyTorch 模型的权重之间的对应关系.

6.3 结果

结果证明了我的想法是对的. 根据这个原理我编写了一个转换程序, 转换得到的模型与原模型的行为完全一致.

转换程序已[开源](https://github.com/ParaN3xus/onnx_map_torch), 我并没有把这个写成一个非常通用的工具, 只是以本文的这个例子走一遍流程, 如果真的有同样的需求应该也能基本复制.

2024-03-05

基于特定端口或协议版本的校园网免流教程

如何在校园网中通过 IPv6, 53 端口等常见授权漏洞实现免流上网

本教程仅供研究学习使用, 请在部署后 24h 内撤销所有更改并恢复原状, 本教程作者对依据本教程行动带来的后果概不负责.

本文的首要假设是你的校园网有某种授权漏洞, 例如 IPv6 或者 53 端口(DNS 所用)不需要鉴权或者不限速等. 若果你的校园网环境没有这些漏洞, 那么请不要尝试本教程.

7.1 成本

虽然打了个 free-ride 的 tag, 但是实际操作并不是免费的, 你仍然需要为一些设备和服务付出代价, 包括:

- 代理服务器: 价格取决于你选择的服务商和具体线路, 配置等. 若可能, 在校内某些地点(如实验室)直接放一台服务器也是可以的.
- 路由器: 至少要允许设置网关.
- 网关: 本地的一台服务器, 如果性能足够, 也可以直接使用支持 OpenWRT 等的路由器.
- 可能被发现的风险(同时违反校规和国家法律)

7.2 最终效果

- 上下行速度达到本方案任意一环中的瓶颈, 对于本人, 这个瓶颈是墙上面板只有百兆.
- 无需付费(指校园网费用).
- 夜间不断网(取决于具体情况).
- 可访问代理服务器可访问的所有网络服务. (如果代理服务器不在校内, 那你可能无法访问校内服务)
- 无限台无线设备连接, 上网无需经过认证或其他任何配置.

7.3 原理解释

某些校园网的鉴权系统非常老旧, 存在诸多漏洞, 比如
- IPv6 不需要鉴权
- 由于需要在无权限时劫持 DNS 请求, 53 端口不需要鉴权

只需要通过这些特殊的端口或协议版本访问某个代理服务器, 然后通过该代理服务器访问互联网, 便可以实现免流上网.

下面的教程以 IPv6 无鉴权的情况为例.

7.4 准备工作

7.4.1 购买路由器

不给出建议.

7.4.2 购买本地网关服务器

理论上来说, 该网关需要满足以下要求:

1. 有线网口速度不成为瓶颈.

2. 能运行 linux, 此方案是完全基于 linux 的.
3. 配置足够高, 能正常运行 xray. 一般来说不会不符合.

如果你想白嫖 Cloudflare Zero Trust 作为代理, 那你的服务器需要是 AMD64 架构, 也即不能使用树莓拍等 ARM 架构的设备, 这是因为 Cloudflare 并不提供 ARM 架构的 WARP 客户端, 而使用第三方 WireGuard 客户端连接 Cloudflare Zero Trust 在如今的环境下几乎不可能成功.

完全可以使用老电脑, 或某些软路由.

7.4.3 购买代理服务器

服务器要满足以下要求:

- IPv6 和 IPv4 双栈公网. 否则要么不能达到解除限速的目标, 要么只能访问支持纯 IPv6 的网站, 而后者在国内几乎不可能满足日用.
  - 或许也可以选择只有 IPv4 的服务器, 然后用 Tunnel Broker 之类的服务获取 IPv6, 但是我觉得这对延迟不友好, 所以没有尝试.
  - 这条要求会缩小选择空间, 很多著名服务商都可能被排除在外.
- 网络配置和硬件配置够好, 价格够便宜. 我们的行为需要有利可图.

7.5 教程

步骤可以大体分为:

1. 配置路由器
2. 配置代理服务器
3. 配置网关

7.5.1 配置路由器

不提供相关指引. 只需要保证路由器能正确访问我们用于访问代理服务器的网络即可.

7.5.2 配置代理服务器

7.5.2.1 自建代理

这一步骤已经被 x-ui 等图形化工具大大简化, 因此这里不再提供详细指引, 只提供一些建议:

- 如果使用了海外代理服务器, 请确保你使用了正确的加密协议
- 如果使用了国内代理服务器, 可以酌情使用 socks5 等更加轻量的协议

7.5.2.2 白嫖 Cloudflare Zero Trust

7.5.2.2.1 注册 Cloudflare 并加入 Zero Trust

可参考此教程的“注册 Cloudflare 帐号并进行设置”部分: 传送门

7.5.2.2.2 进行一些设置

在 Zero Trust 面板中, 进入 Settings, WARP Client, Device settings, Default, 点击最后的三个点, Configurate.

打开 Mode swtich, Auto connect, Service mode 选择 Proxy, 点击 Save Profile 保存.

7.5.2.2.3 配置 WARP Client

7.5.2.2.3.1 安装客户端

请参考官方教程: [传送门](https://pkg.cloudflareclient.com/)

7.5.2.2.3.2 加入团队

```
warp-cli teams-enroll 你的团队域
```

然后终端中会出现一个地址, 本地访问这个地址, 并在其中按照指引登录.

```
warp-cli teams-enroll-token 你的token
```

其中 token 可在浏览器登陆后的页面里打开开发者工具查看, 直接复制 Open Cloudflare WARP 按钮指向的那个地址即可.

```
warp-cli status
```

应当输出

```
Status update: Connected
Success
```

至此, WARP Client 配置完成.

7.5.3 配置网关

可参考教程: 传送门

关于教程中的 Xray 配置部分, 你只需要进行客户端的配置, 本人的配置文件如下, 请将信息修改为你本人的信息后参考使用.

```
{
  "log": {
    "loglevel": "warning"
  },
  "inbounds": [
    {
      "tag": "all-in",
      "port": 12345,
      "protocol": "dokodemo-door",
      "settings": {
        "network": "tcp,udp",
        "followRedirect": true
      },
      "sniffing": {
        "enabled": true,
        "destOverride": [
          "http",
          "tls",
          "quic"
        ]
      },
      "streamSettings": {
        "sockopt": {
          "tproxy": "tproxy"
        }
      }
    },
    {
      "port": 10808,
      "protocol": "socks",
      "sniffing": {
```

```
      "enabled": true,
      "destOverride": [
        "http",
        "tls",
        "quic"
      ]
    },
    "settings": {
      "auth": "noauth",
      "udp": true
    }
  }
],
"outbounds": [
  {
    "tag": "proxy",
    "protocol": "vless",
    "settings": {
      "vnext": [
        {
          "address": "填写代理服务器IPv6地址",
          "port": 填写代理服务器入站节点端口,
          "users": [
            {
              "id": "按照节点配置页面填写",
              "alterId": 0,
              "email": "按照节点配置页面填写",
              "security": "auto",
              "encryption": "none",
              "flow": ""
            }
          ]
        }
      ]
    },
    "streamSettings": {
      "sockopt": {
        "mark": 255
      },
      "network": "tcp",
      "security": "reality",
      "realitySettings": {
        "serverName": "按照节点配置页面填写",
        "fingerprint": "chrome",
        "show": false,
        "publicKey": "按照节点配置页面填写",
        "shortId": "",
        "spiderX": ""
      }
    },
    "mux": {
      "enabled": false,
      "concurrency": -1
    }
  },
  {
```

```
      "tag": "direct",
      "protocol": "freedom",
      "settings": {
        "domainStrategy": "UseIP"
      },
      "streamSettings": {
        "sockopt": {
          "mark": 255
        }
      }
    },
    {
      "tag": "block",
      "protocol": "blackhole",
      "settings": {
        "response": {
          "type": "http"
        }
      }
    },
    {
      "tag": "dns-out",
      "protocol": "dns",
      "streamSettings": {
        "sockopt": {
          "mark": 255
        }
      }
    }
  ],
  "dns": {
    "hosts": {
      "domain:googleapis.cn": "googleapis.com",
      "dns.google": "8.8.8.8"
    },
    "servers": [
      "https://1.1.1.1/dns-query",
      {
        "address": "119.29.29.29",
        "domains": [
          "geosite:cn"
        ],
        "expectIPs": [
          "geoip:cn"
        ]
      },
      "https://dns.google/dns-query",
      "223.5.5.5",
      "localhost"
    ]
  },
  "routing": {
    "domainMatcher": "mph",
    "domainStrategy": "IPIfNonMatch",
    "rules": [
      {
```

```
      "type": "field",
      "domain": [
        "geosite:category-ads-all"
      ],
      "outboundTag": "block"
    },
    {
      "type": "field",
      "inboundTag": [
        "all-in"
      ],
      "port": 123,
      "network": "udp",
      "outboundTag": "direct"
    },
    {
      "type": "field",
      "inboundTag": [
        "all-in"
      ],
      "port": 53,
      "network": "udp",
      "outboundTag": "dns-out"
    },
    {
      "type": "field",
      "ip": [
        "119.29.29.29",
        "223.5.5.5"
      ],
      "outboundTag": "proxy"
    },
    {
      "type": "field",
      "protocol": [
        "bittorrent"
      ],
      "outboundTag": "proxy"
    },
    // 我不好说
    {
      "type": "field",
      "ip": [
        "geoip:cn"
      ],
      "outboundTag": "proxy"
    },
    {
      "type": "field",
      "domain": [
        "geosite:cn"
      ],
      "outboundTag": "proxy"
    },
    // end of 我不好说,
    {
```

```
        "type": "field",
        "ip": [
          "geoip:private",
          "填写代理服务器IPv6地址",
          "填写代理服务器IPv4地址"
        ],
        "outboundTag": "direct"
      },
      {
        "type": "field",
        "ip": [
          "1.1.1.1",
          "8.8.8.8"
        ],
        "outboundTag": "proxy"
      },
      {
        "type": "field",
        "domain": [
          "geosite:geolocation-!cn",
          "domain:googleapis.cn",
          "dns.google"
        ],
        "outboundTag": "proxy"
      }
    ]
  }
}
```

该教程中有大量代表局域网网段或主路由、代理服务器的 IP 地址, 请在配置时注意进行替换.

注意在该教程中, 最后的局域网设备上网部分可以使用方案二, 直接在路由器配置页面中设置网关即可.

7.6 最终测试

国内测速站点: [传送门](https://test.ustc.edu.cn/)

国外测速站点: [传送门](https://www.speedtest.net/)

7.7 总结

祝每一位读者获得更好的校园网上网体验.