ZAI-DNNOD(MaxMargin-ObjectDetector)建模指南

文档版本：1.1
更新时间：2019-2
文档作者：qq600585

目录

概念：

DNN-OD 也叫 MaxMargin-ObjectDetector，它是从 ObjectDetector(HOG+silde window)基础上发展而出的 GPU 版本 OD，它对 Object 的检测比 ObjectDetector 更加先进。

MMOD 可以使用较少图片，达到很好的效果。譬如，OD 建模需要 5000 张样本图片，MMOD 则只用 100 张即可达到 OD 的效果，因为 MMOD 是基于 GPU 在训练，我们可以在训练前做很多内容重构的预处理，诸如图片内容变形，图片内容受到光线变化影响，这些样干就让 MMOD 具备更强大的检测能力。

MMOD 可以检测不同的对象并且给出对象标签：当我们在训练 MMOD 时定义好标签，MMOD 检测出来的对象就会有标签。MMOD 具备一定的模式识别能力但不是专门解决模式识别问题的方案，专业的模式识别是 RNIC 技术体系。

MMOD 依赖于 GPU 工作，OD 依赖于 CPU 工作。
MMOD 检测和训练图片都需要显存和内存足够大，OD 只需要内存足够大。

MMOD 是 OD 的替代方案，假如有 GPU 硬件支持，MMOD 各项指标都会优于 OD。

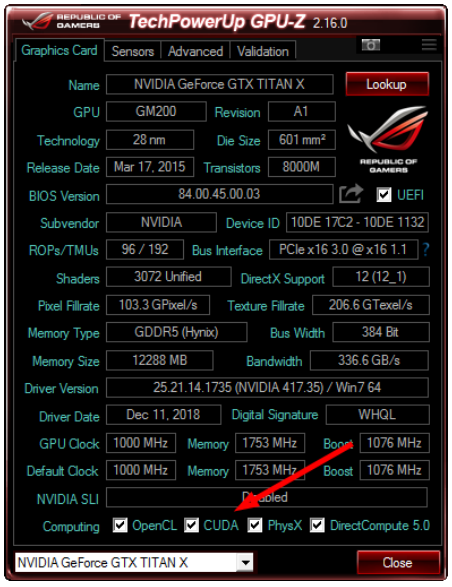
在需要做 SP 检测的地方，比如人脸，工业，传统 OD 往往要比 MMOD 有效，MMOD 会检测到重叠和变形的目标对象，传统 OD 只会检测拥有完整表面信息的目标对象。

而当我们需要做行人流统计检测，车辆统计检测，车牌检测，MMOD 的重叠和变形识别能力，MMOD 会优于传统 OD 检测器。

MMOD 建模准备工作

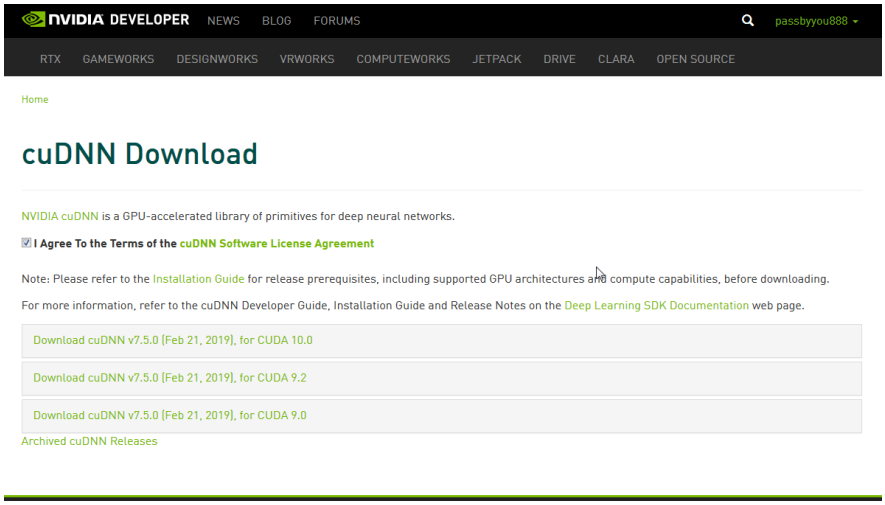
Cuda 准备工作

需要 cuda 硬件设备，参考文档，zAI 开发设备撰机指南.pdf
另一个是 gpu-z 检测器，它有两个版本，一个是华硕板皇的黑版，一个是普通版。主要差别是皮肤不一样。
https://www.techpowerup.com/download/techpowerup-gpu-z/
在 gpu-z 中确保显卡是支持 cuda 的



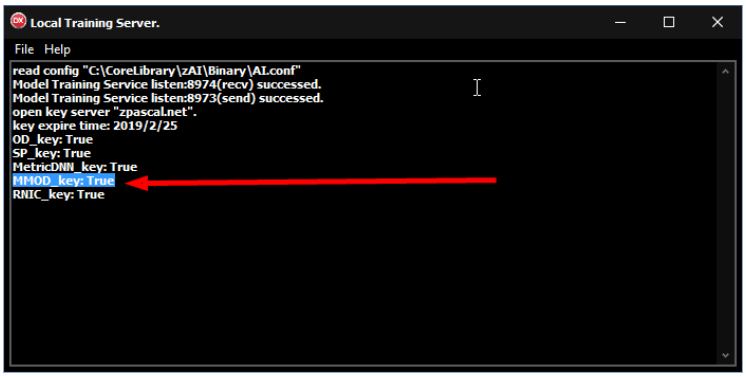
cuda 需要到 nvidia 的官方网站下载驱动，zAI 的 cuda 引擎使用 nvcc+cuda sdk10 构建，也支持 cuda9.2，如果是 cuda9.0 或更低的版本是不支持的。
https://developer.nvidia.com/cuda-downloads

另外我们还需要单独安装一个和 cuda 对应的 cuDNN 库，这个库比较大，且需要和 cuda 配套。提示：我们下载 cudnn 需要一个 nvidia 开发者账号，过程很简单，按提示操作即可。
https://developer.nvidia.com/rdp/cudnn-download

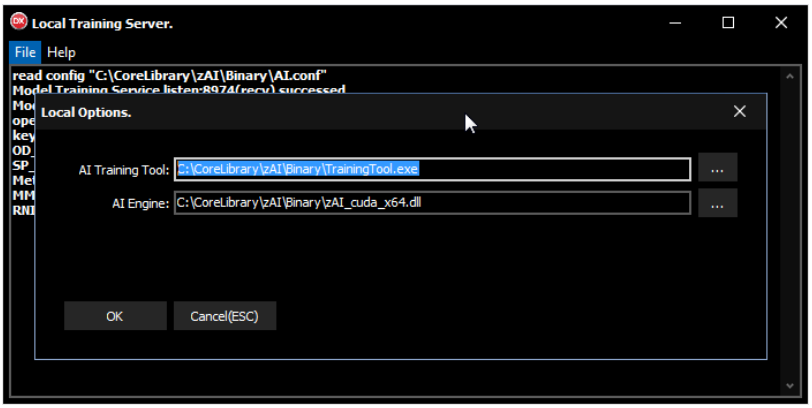


LocalTrainingServer 准备工作

确保 zAI 的 key 是被激活的，打开 LocalTrainingServer，确保 mmod_key:True

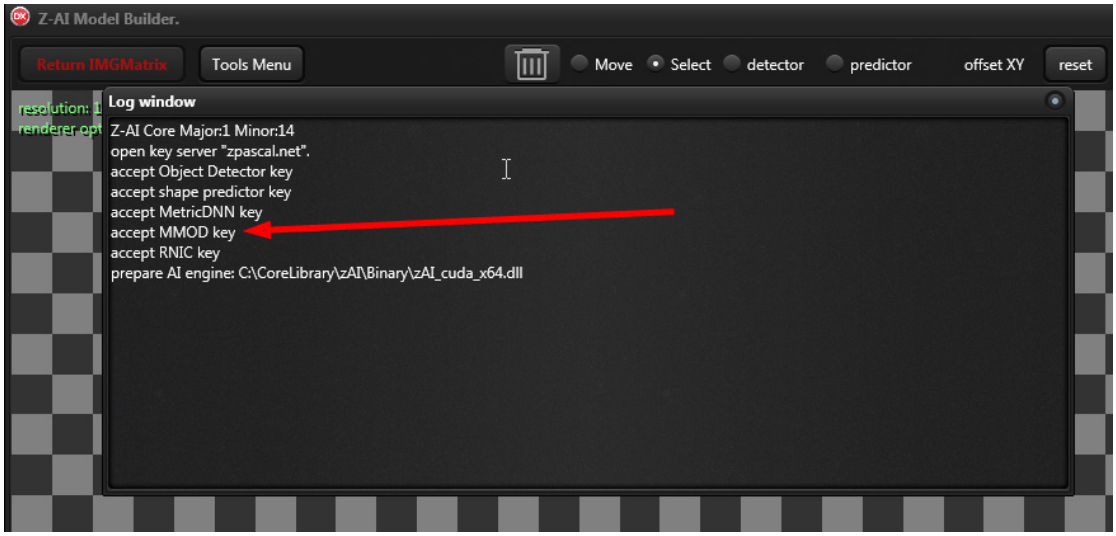


在 Training Server 的 Options 设置中，确保 TrainingTool 和 ZAI 的 Cuda 引擎都是正确的



接下来，打开 z_ai_model

如果 key 可以使用，会提示 accept MMOD key



MMOD 的策略一建模

第一步，准备足够需要让 MMOD 识别的目标样本，上万张图片也不过分
第二步，将目标样本分为两批，A 部分为少量抽样采集的样本，B 为主样本
第三步，使用 z_ai_model 对 A 部分进行手动建模
第四步，用 MMOD 训练 A 部分
第五步，用训练完成的 A 部分样本的 MMOD 模型，去标注 B 样本集
第六步，修正 B 样本集
第七步，用 MMOD 训练 B 样本集

MMOD 的策略二建模

第一步，准备足够需要让 ODM 识别的目标样本，上万张图片也不过分
第二步，将目标样本分为两批，A 部分为少量抽样采集的样本，B 为主样本
第三步，使用 z_ai_model 对 A 部分进行手动 ODM 建模
第四步，用 ODM 训练 A 部分
第五步，用训练完成的 A 部分样本的 ODM 模型，去标注 B 样本集
第六步，修正 B 样本集
第七步，用 MMOD 训练 B 样本集

两种建模策略的总结

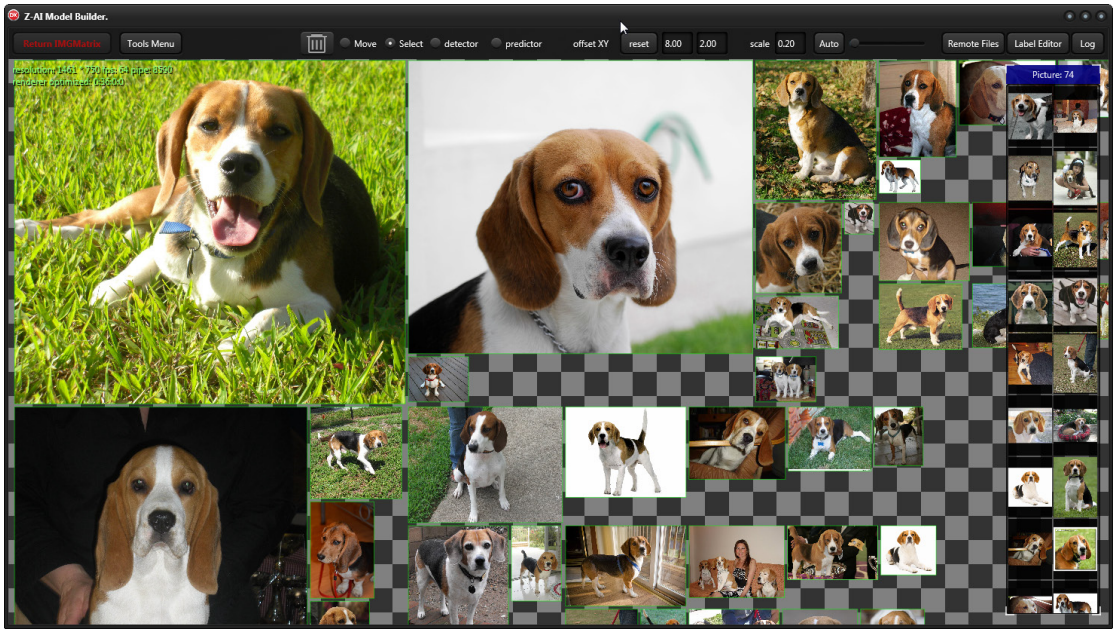
两种建模方法分别是 MMOD+ODM，他们的共同点都是，检测到对象以后，会反馈一个标签。MMOD 的机制设计就是让它检测对象并且用 Label 告诉你这个对象是什么.

策略一使用 GPU 抽样预建模，策略二则使用 CPU 抽样预建模。
待预建模完成后，再用预建模的结果去自动化的标注大规模数据集。

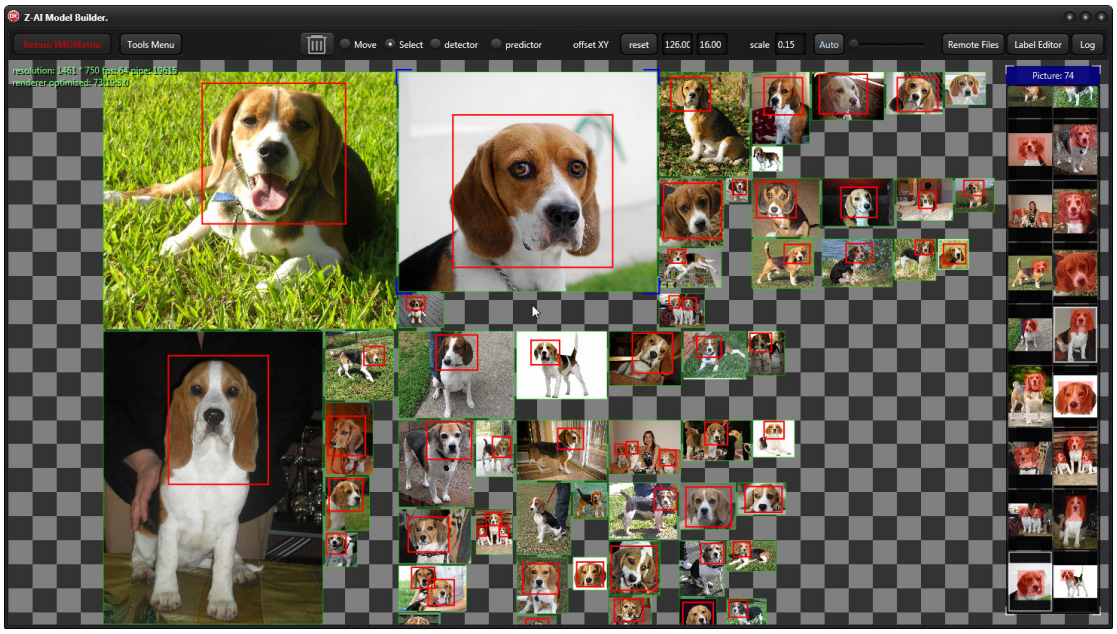
抽样导入我们的第一批图片

一般来说，图片分辨率在 1800*1800 以下之间都不用做什么处理
如果图片尺寸过大，我们的内存+显存将会告急，我们需要做一下减肥处理，参考文档，使用 PS 自动脚本批处理.pdf

本文以巴哥为例，导入了一批已做过剪裁的抽样图片

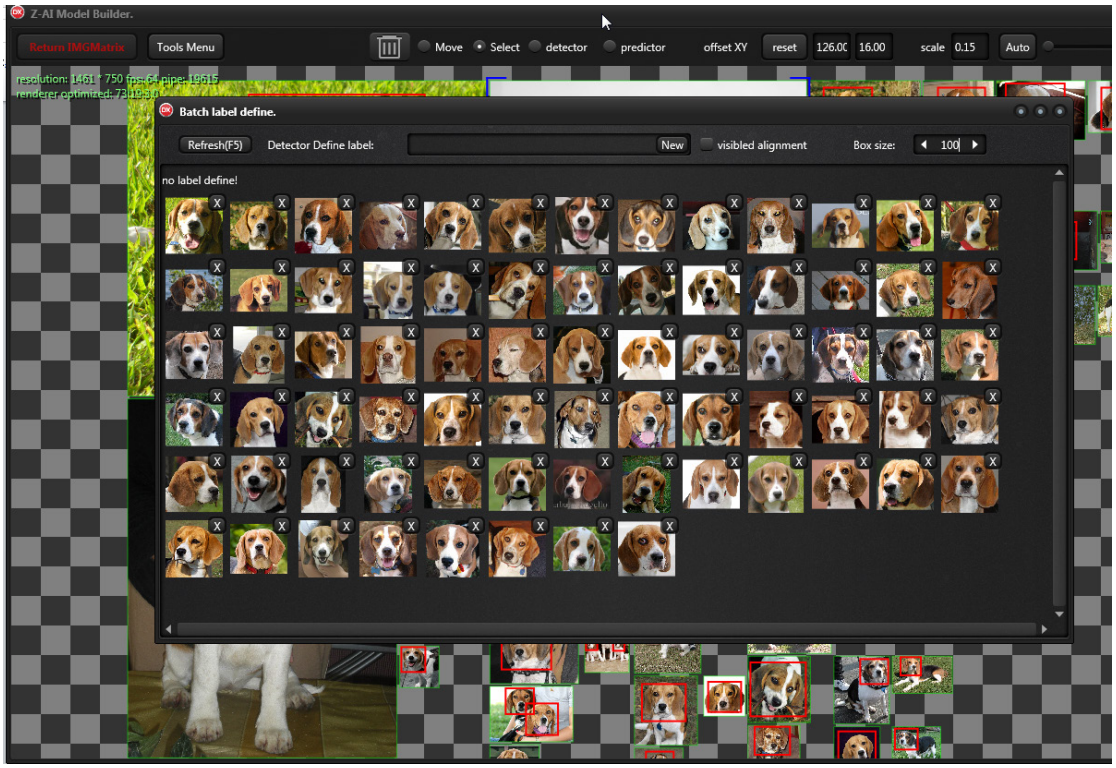


开始画狗头框体

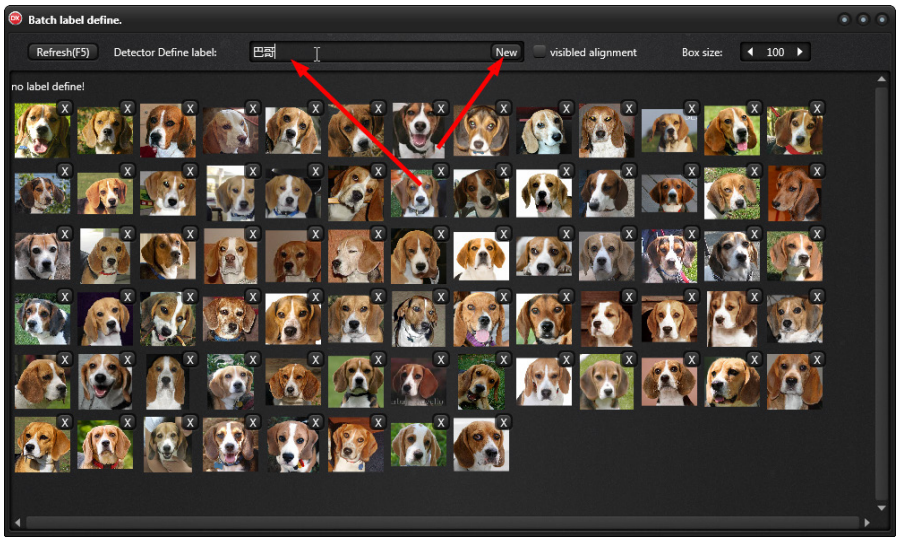


使用标签编辑器检查框体错误

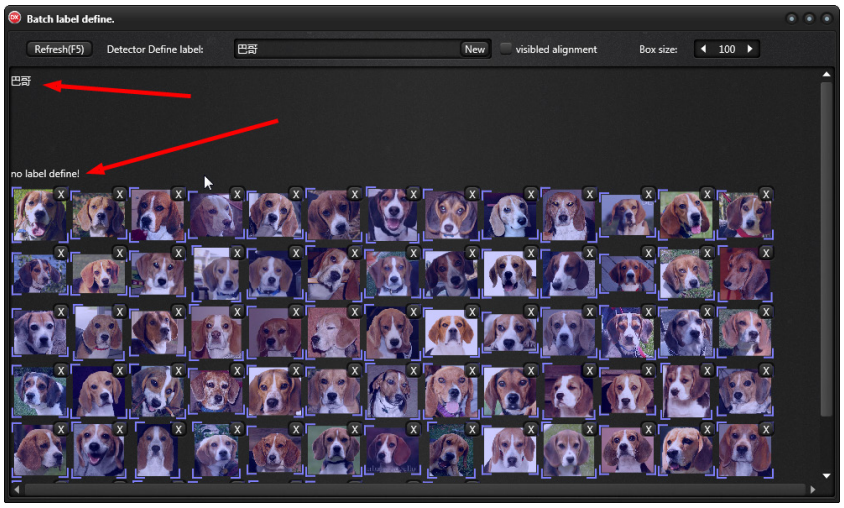
检查有没有框错的，有就干掉



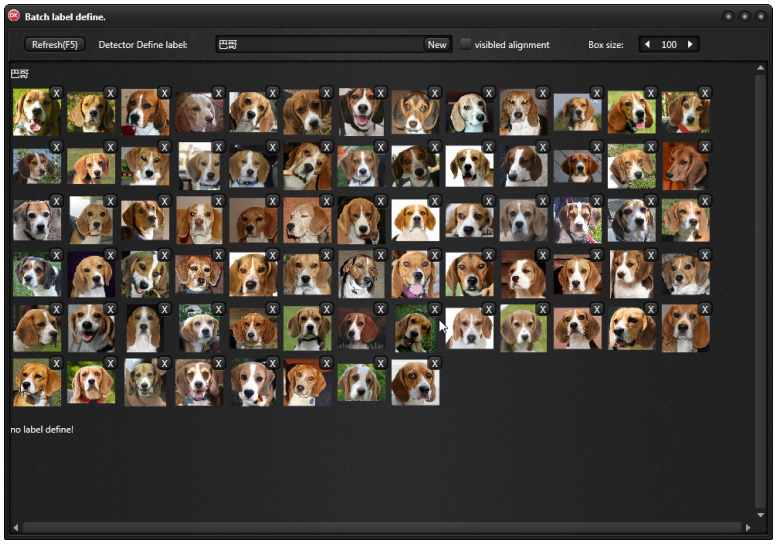
使用标签编辑器定义一个巴哥标签



然后将 no label define!拖动到巴哥标签上

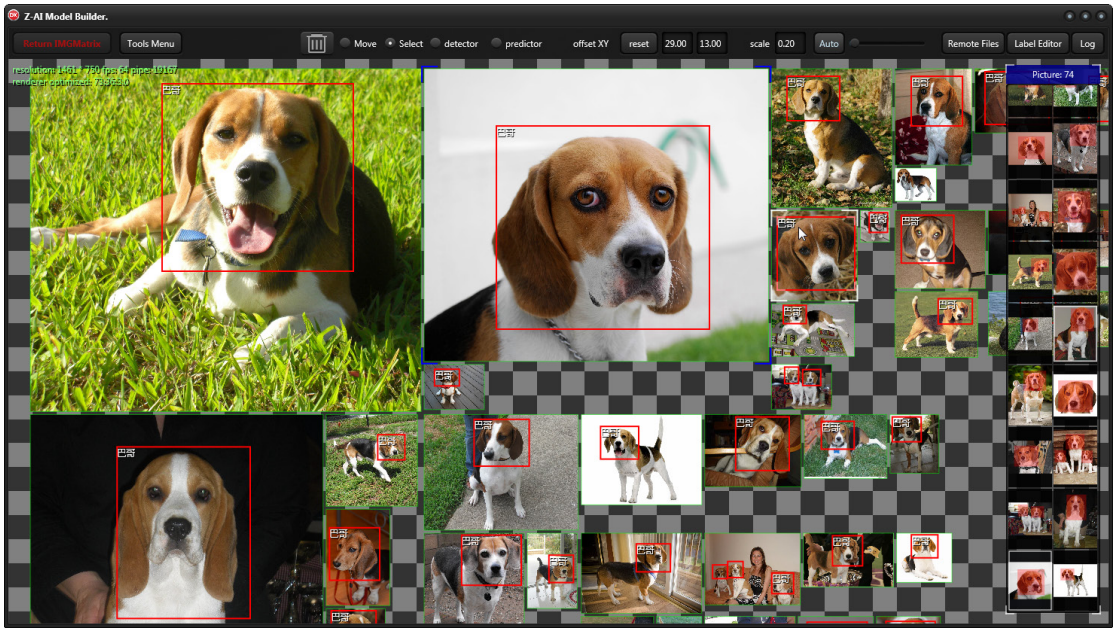


完成后关闭标签编辑器



巴哥的狗头已被标记完成

标记完成后，我们在编辑器就能看到巴哥的标记

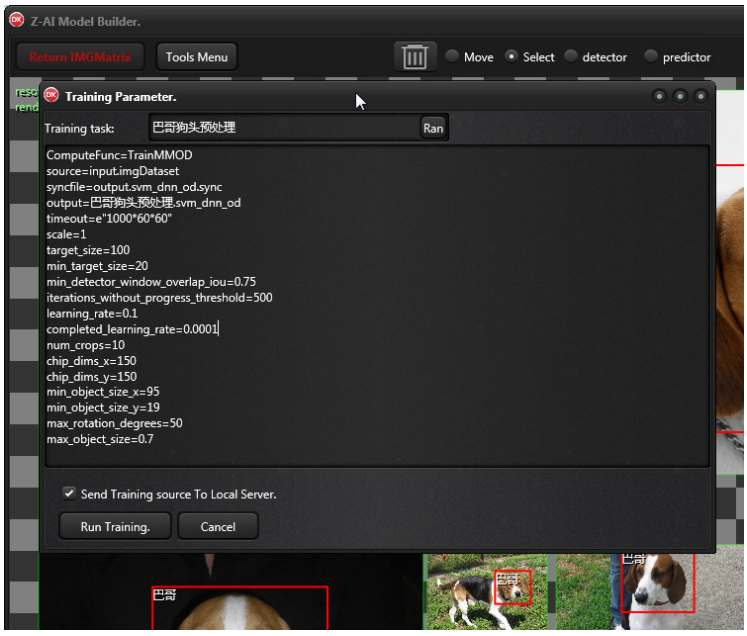


使用 MMOD 训练巴哥狗头

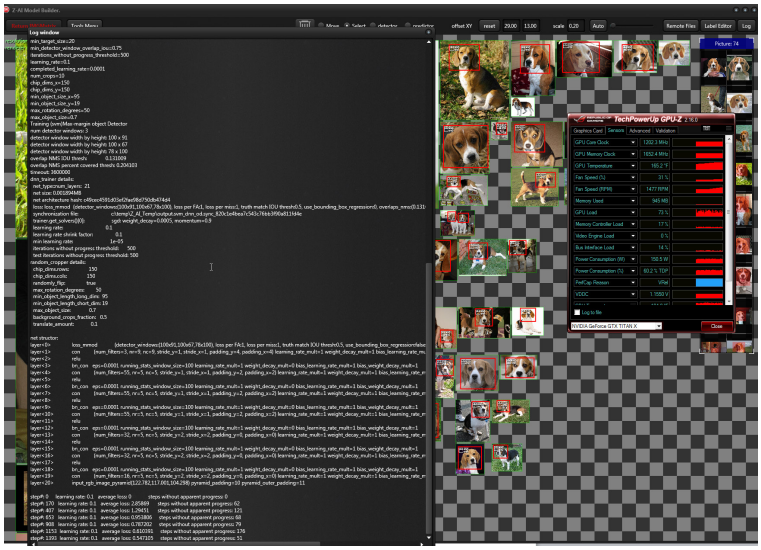
MMOD 因为使用了 GPU，训练参数很复杂，这些参数叫做超参数，他们的含义很隐晦，往往堆积大量的数学公式来解释，使用正常的语言很难解释清楚，对超参数最好的理解是通过 *理解->实际训练->加深理解* 三个步骤来。

理解 MMOD 的训练超参数，请参考 zAI 的 Demo:DNN-OD，在程序源代码中已解释了这些超参数的含义，或许你可以做一个训练参数的学习笔记，因为这是标准机器学习的 resnet 范式。

训练参数如果给错，可能会需要几十个小时来训练，或则训练出来的模型达不到预期精确度，而恰到好处的超参数则会让训练事半功倍。



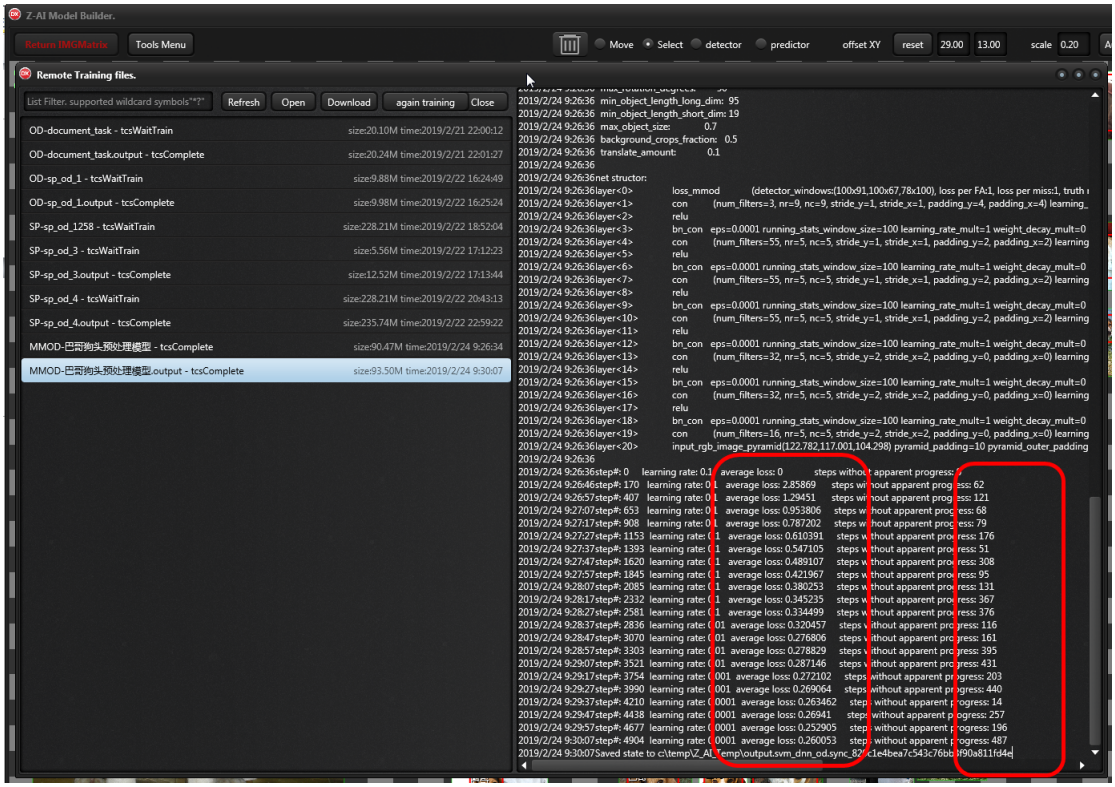
MOOD 的训练是纯 gpu，MMOD 模型的训练往往极度耗时，我们训练也不会像 OD 那样一次性就成功。



把训练发给服务器以后，我们就可以关闭 z_ai_model 了，耐心等待结果

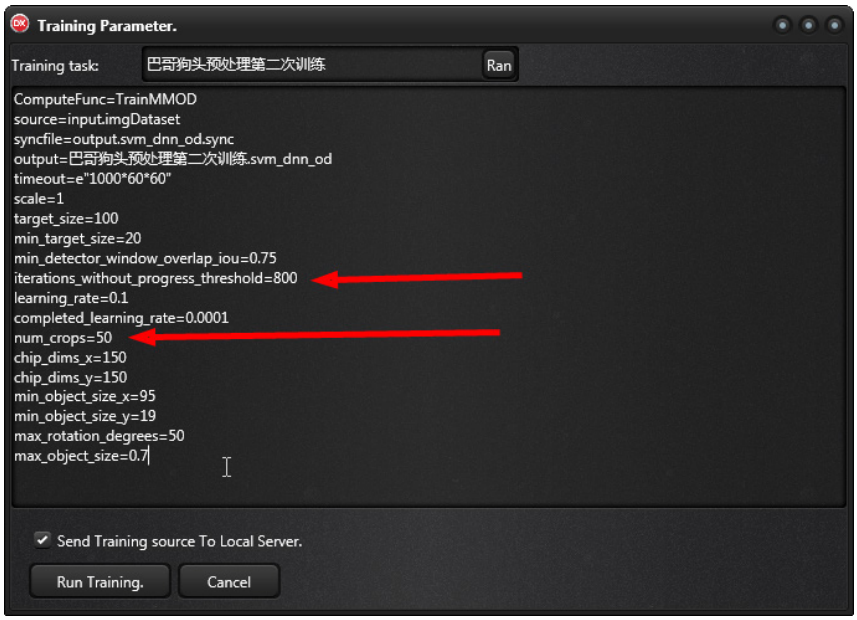
5 分钟以后，服务器训练完成

我们通过 remote files，可以查看到巴哥的训练结果
通过 average loss 精度，可以判断，这个巴哥识别率大概只有 80%，这是不合格的模型
这一次，我们训练就算失败，问题出在训练参数

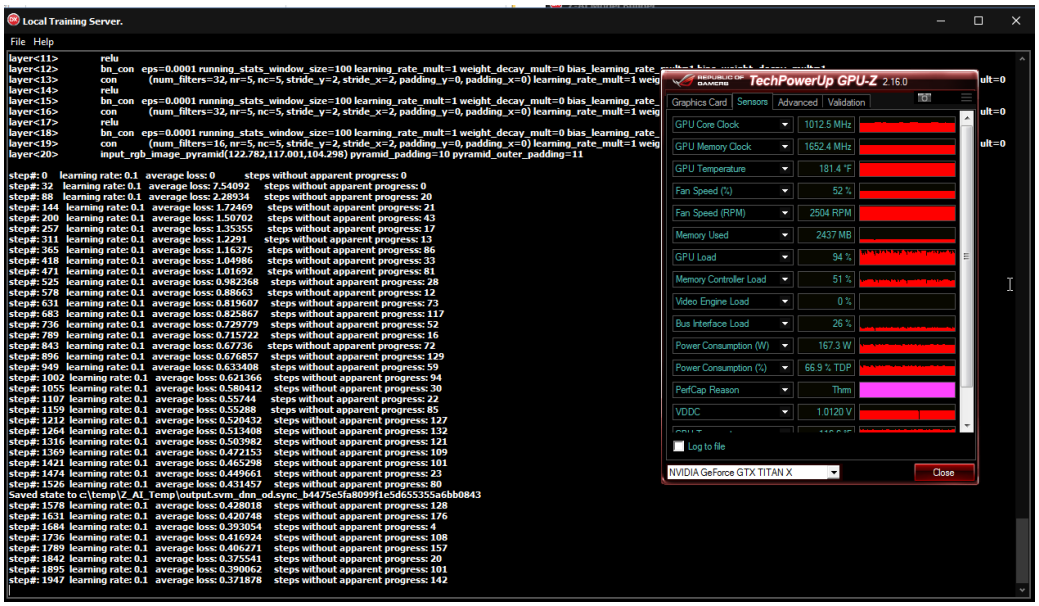


第一次 MMOD 训练精度不合格，我们需要调整一下超参

我将*无效步数调整为 800*，关于这个参数的解释，请参考 demo:DNN-OD
我将*裁剪次数改成 50*，关于这个参数的解释，请参考 demo:DNN-OD

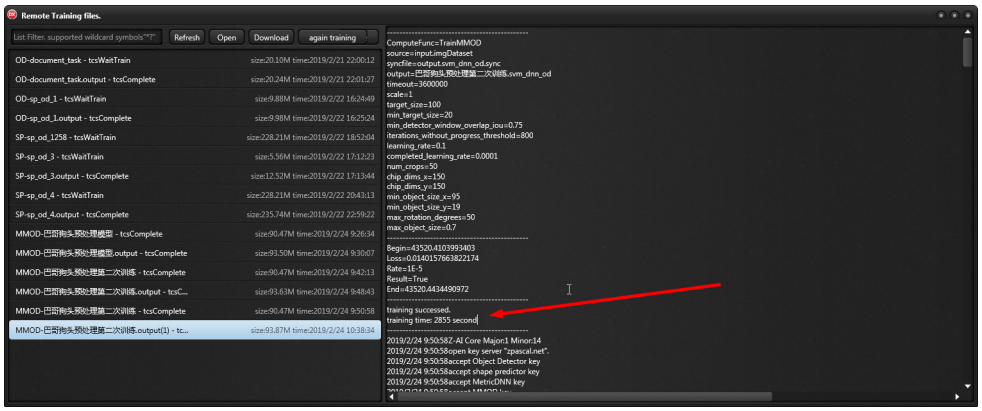


然后，再训练一次
在服务器训练中，*我们可以看到 average loss 正在逼近更小，无效步数大约在 200 内。通过前 5 分钟的逼近观察，结合我的给出了 timeout=1 小时的参数设置，狗头应该会训练 2-3 次左右*（因为 1 小时以后，训练会因为 timeout 强制结束，并且保存进度，然后我需要使用 again training 模式再次训练）。
gpu-z 的使用率 80%，cpu 使用率 20%，硬件是不会罢工的。现在，设定一下我们手机的闹钟，对 siri 说，嘿，siri 帮我定一个一小时以后的提醒。我们去找儿子玩一会再回来吧。

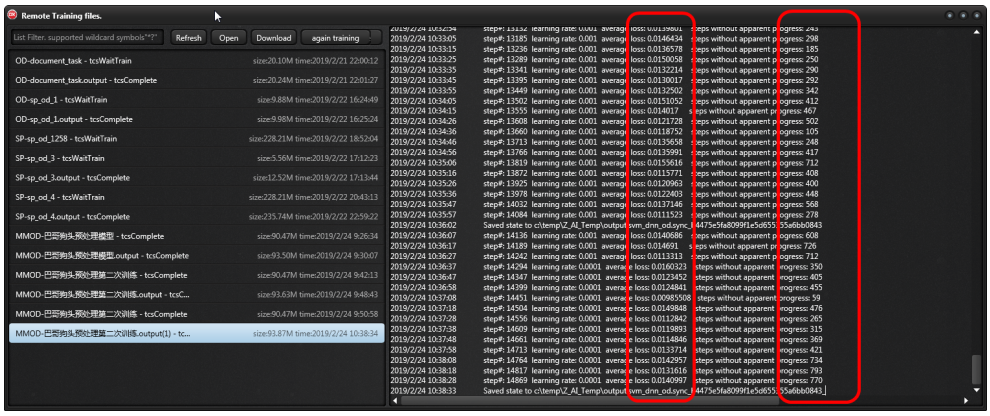


一个小时后，第二次训练完成

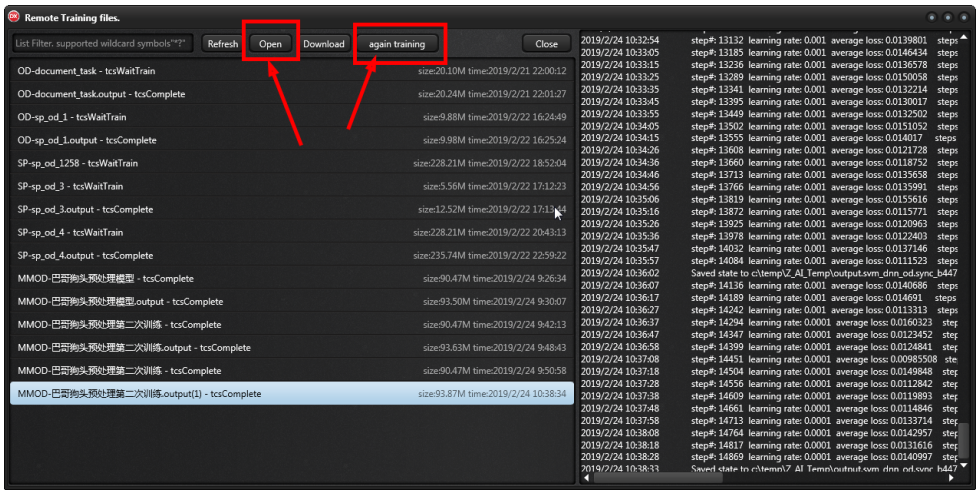
我们打开 remote files，查看我们最后训练的任务
本次任务的计划 timeout 时间是 1 小时，3600 秒，实际训练完成花了 2800 秒，大概 40 分钟左右。
我猜测的是 2-3 个小时，而实际训练只花了 40 分钟，这是因为人品好吧。



我们再来看一下精度结果，大概 99%左右的精确度，好了，这个模型可以用了



我们 open 它
提示：如果训练是因为 timeout 退出，我们使用 again training 可以再次设置参数继续上次未完成的训练，不用从头开始。



在 FilePackageTool 中将训练结果导出来

Param.txt，我们刚才输入的训练超参，可以双击打开

Iput.imgDataset，这个 90M 的大文件，是我们输入的训练的数据集，导出以后可以直接用 z_ai_model 打开编辑

巴哥狗头预处理第二次训练.svm_dnn_od，这是我们的训练结果，这是 mmod 模型，我们需要将它导出到桌面

output.svm_dnn_od.sync，同步文件，因为 gpu 训练时间都非常久，zAI 每隔 5 分钟会生成一次同步文件，同步文件可以用于恢复训练状态，同步文件只在 zAI 中的 gpu 技术体系中才有，在 ZAI 的 CPU 技术体系中，没有这种东西，因为 CPU 不适合做大规模训练。

Result.txt，训练结果报告，可以双击打开，下面是 Result.txt 的内容说明

- begin 开始物理时间
- loss 是丢失率，1.0-loss=准确率
- rate=1e-5，等同于 0.0001 我们的预期就是 0.0001，学习频率精度，每一次无效步数达到条件，频率就会迭代
- Result=True，表示训练成功
- End 结束的物理时间

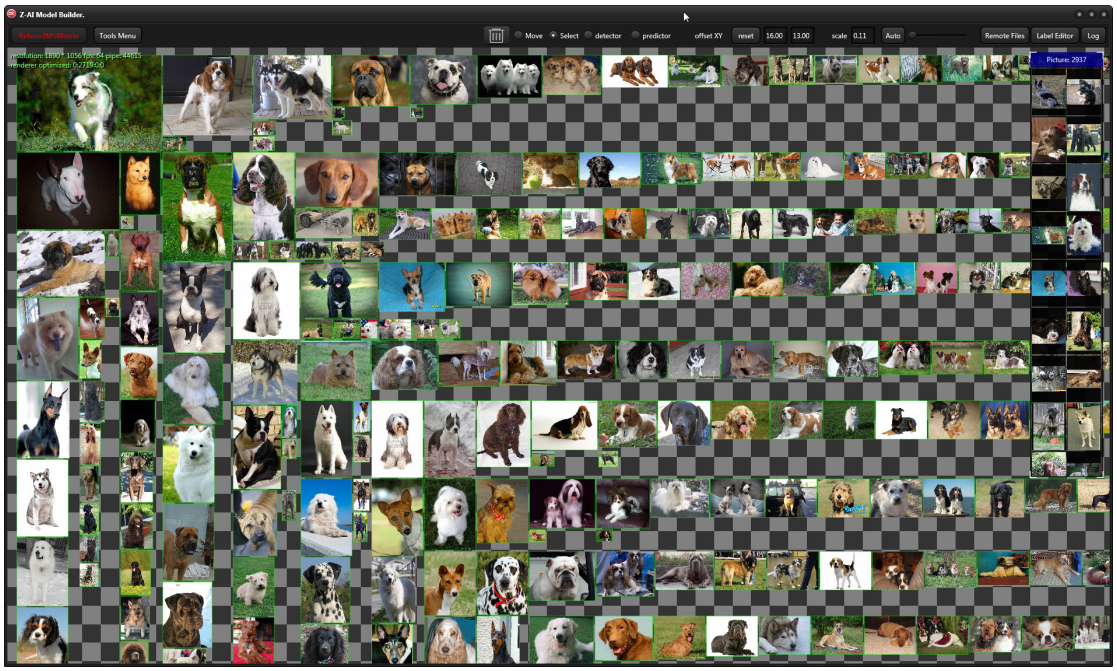
Log.txt，日志，有时间前缀
Engine_log.txt，引擎日志，无时间前缀

Training_steps.dat，步数信息，这是 gpu 训练独有的数据结构，它记录了每一次步数的训练状态，使用 ToneStepList 类读取，需要绘制训练曲线做统计+分析，那么数据源就是它。

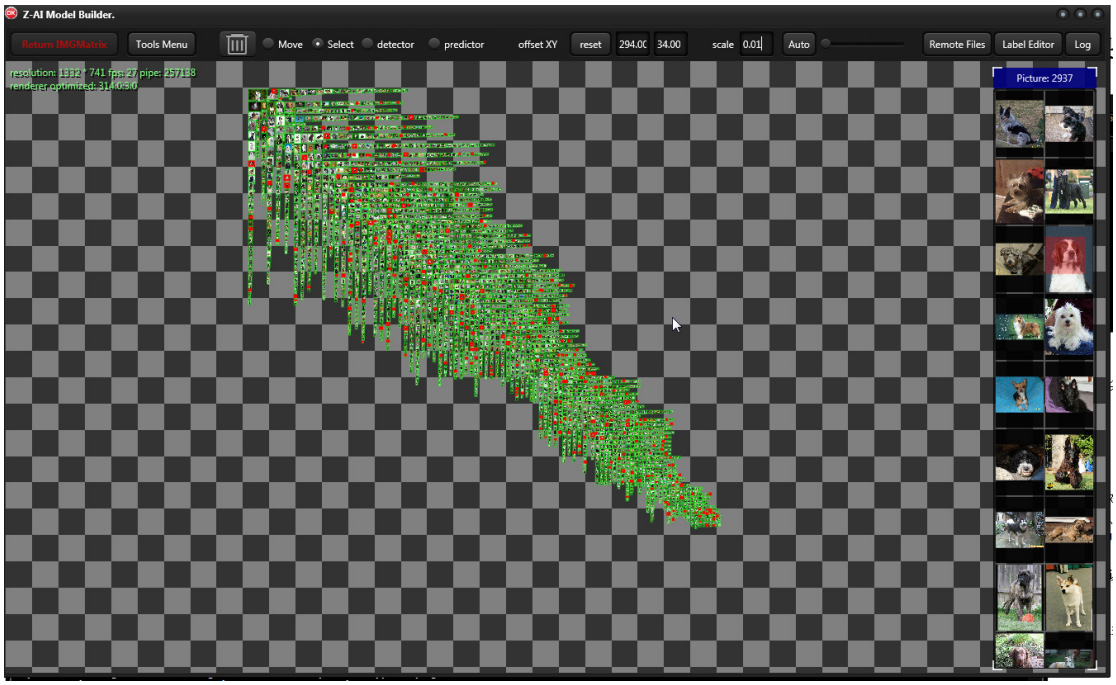
LastOutput.txt，输出文件列表

开始大量制作狗头标注

这次我们导入了 3000 张小狗的图片，内存使用了接近 15G（原数据集 18000 张，需要 64G 内存才能训练）

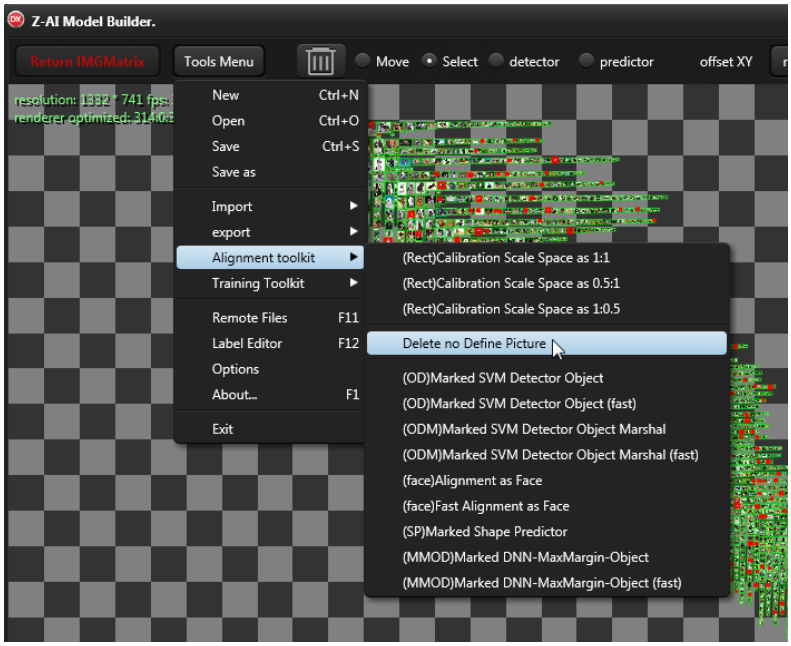


使用 巴哥狗头预处理第二次训练.svm_dnn_od 做自动标注
标注完成后，我们会看见若干红色标注框，因为图片太多，缩小了 100 倍后才能鸟瞰

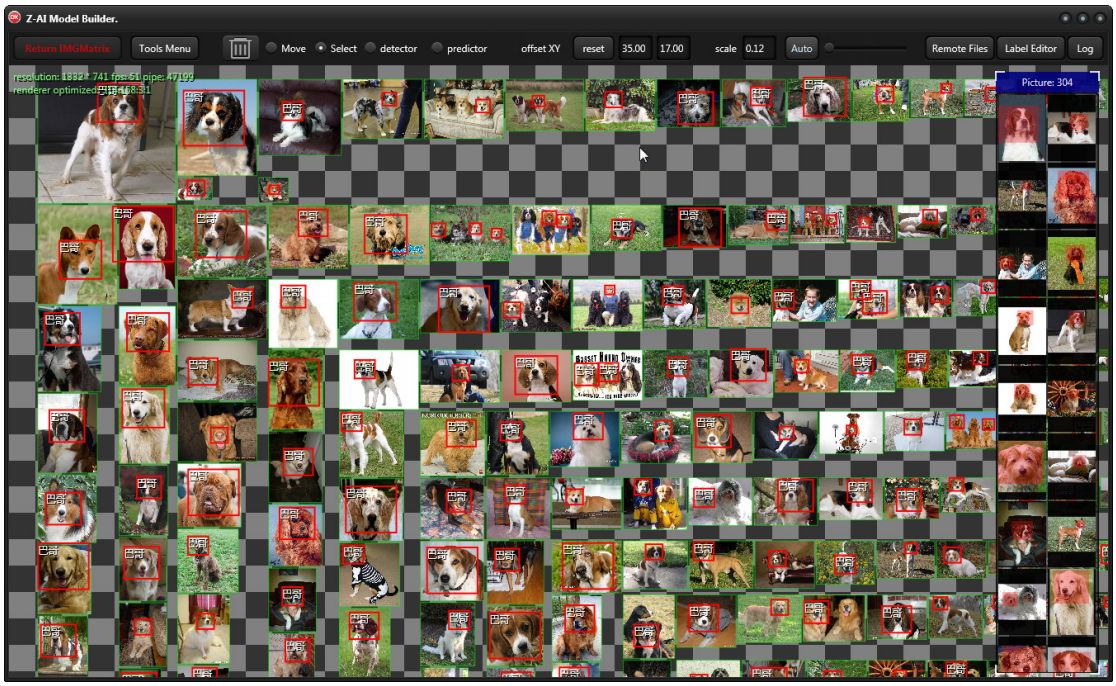


干掉没有被标注出来的狗头

使用工具箱菜单干掉它们

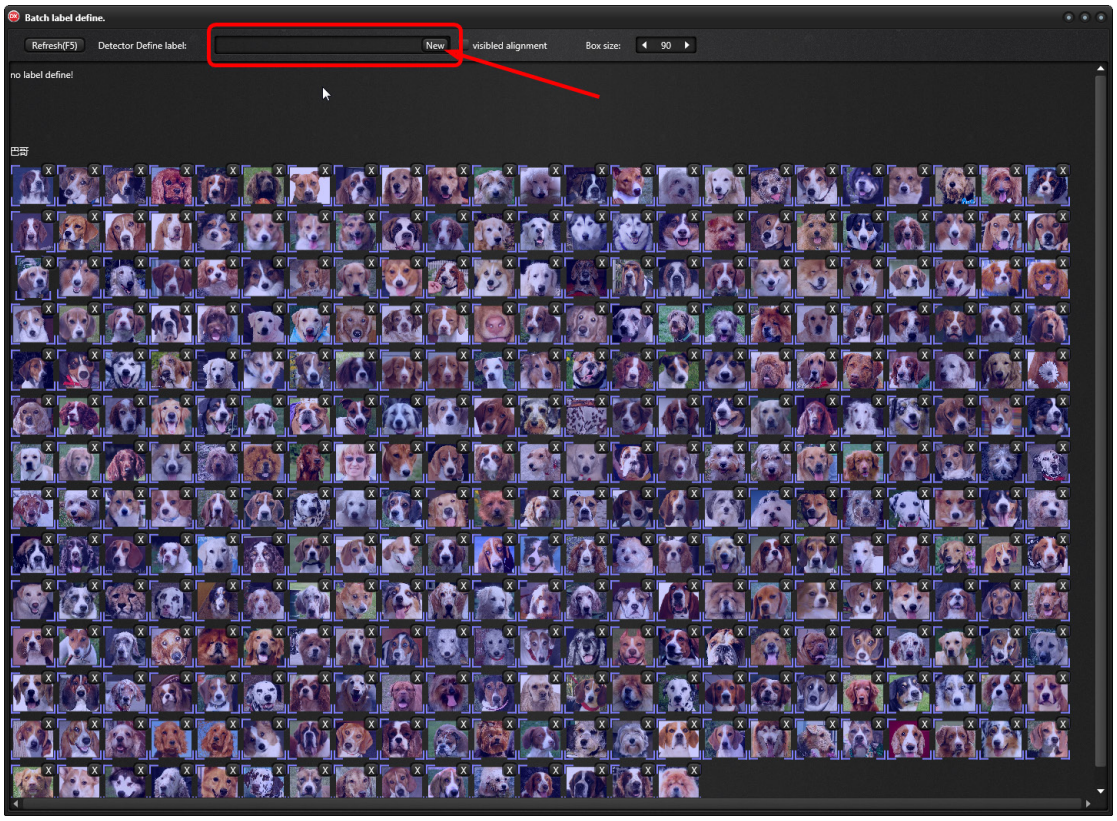


干掉后，做一个排序，我们看到，很多非巴哥的狗头被标注了巴哥

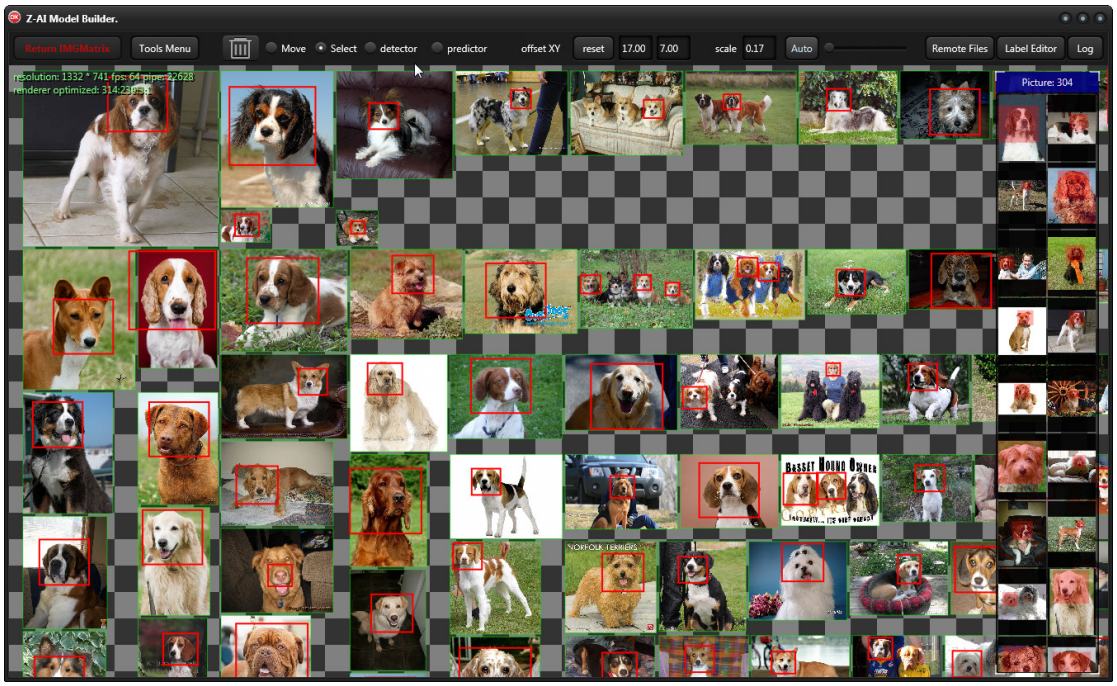


使用 label editor 将巴哥狗头标签干掉

什么都不用输入，直接点 new，会创建一个空标签
让将巴哥标签拖动到 no label define!上面

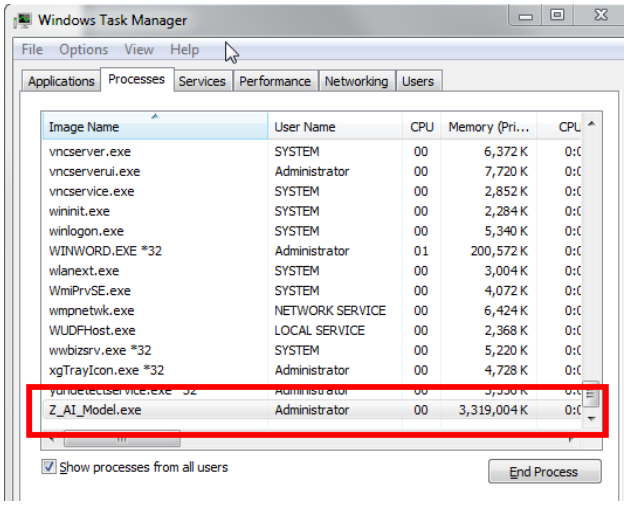


回到编辑器



准备再次训练这 300 张新标注的狗头

我们先查一下内存使用量



300 张图片足足使用了我们 3G 的内存。光是存储和服务器交换都要需要一些时间了。

这时候，如果是低配服务器和工作站，需要使用手动训练这 300 新标注狗头

在实际使用中，手动训练两种方式，

1， 准备好训练数据集，然后使用 TrainingTool.exe 在命令行训练

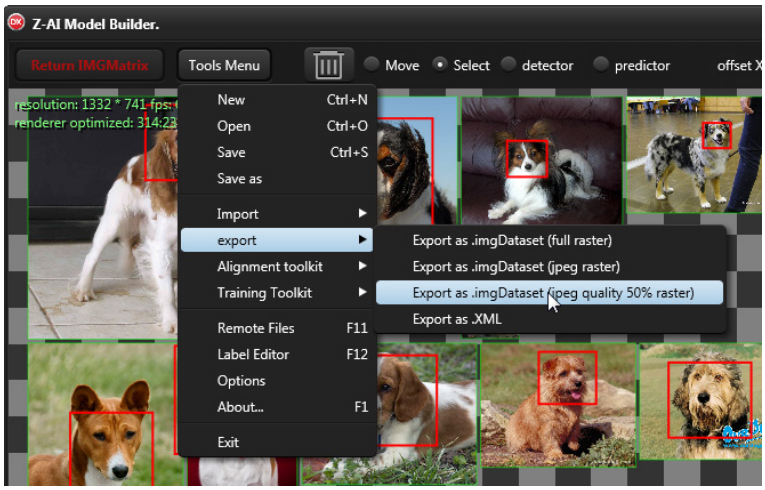
2， 参照 zAI 的 Demo：DNN-OD，直接使用 Delphi 编程来进行训练

由于大规模数据集处理根本不可预测，手动训练将会是使用 zAI 建模的必须环节

如果开发设备的配置足够强劲，可以无视 3G 内存开销，直接发给服务器训练

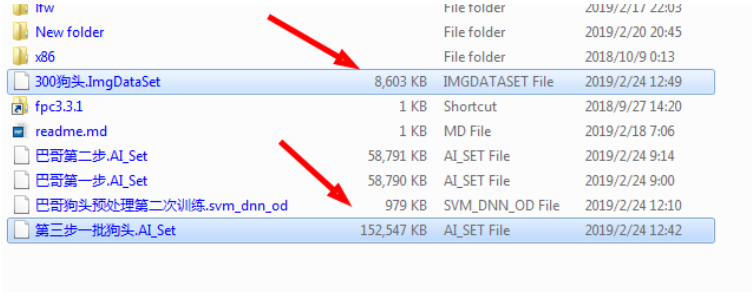
低配设备：手动训练 300 新标注狗头

首先，我们将画质降低一半，以 jpeg 模型存储成 300 狗头.imgDataset



完成后，我们来对比一下

无损光栅 150M，降画质 50%后 8M，文件存储 size 有 20 倍的差异



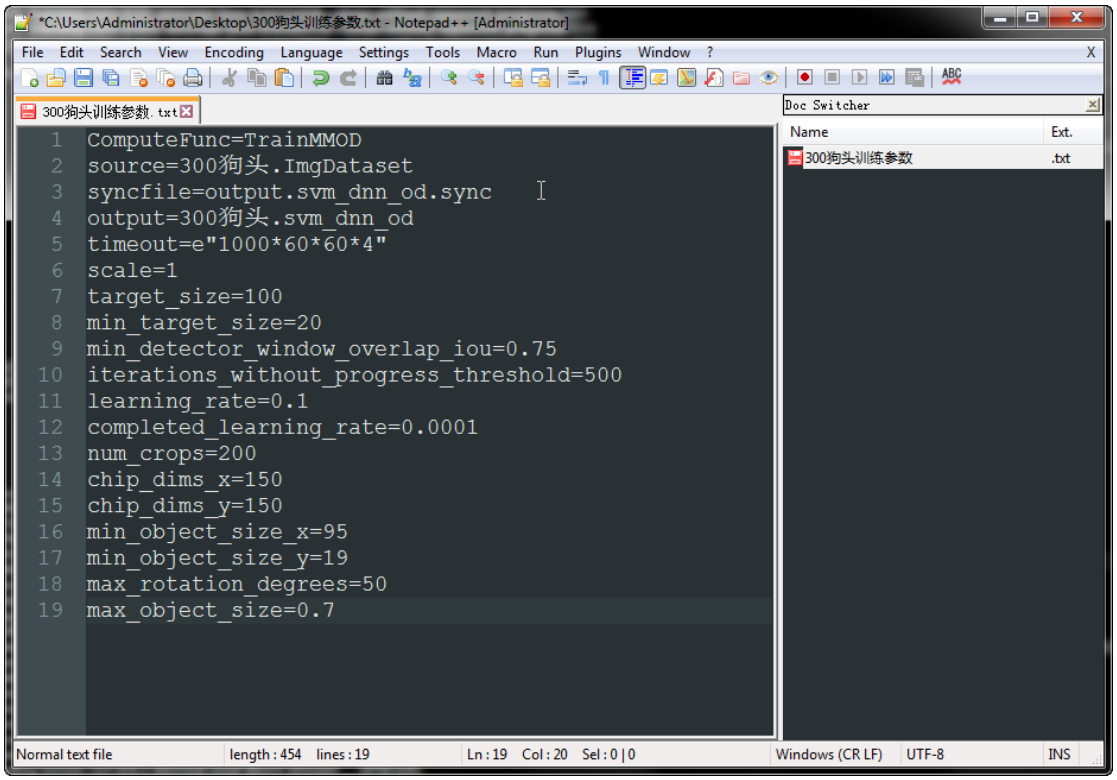
使用 TrainingTool.exe 在命令行手动训练

TrainingTool 运作的中心思路是，1 个文件输入，训练，1 个文件输出

TrainingTool 支持的输入文件格式是.ox，我们使用 FilePackageTool 制作

用 FilePackageTool 制作 TrainingTool 的输入数据

新建一个文本文件，命名为，”300 狗头训练参数.txt”
内容就用 MMOD 的文本参数，我给的 num_corps=200 要求 gpu 至少 10G 显存，如果配置不够，可以将该参数给 100



几个关键说明

ComputeFunc=训练函数

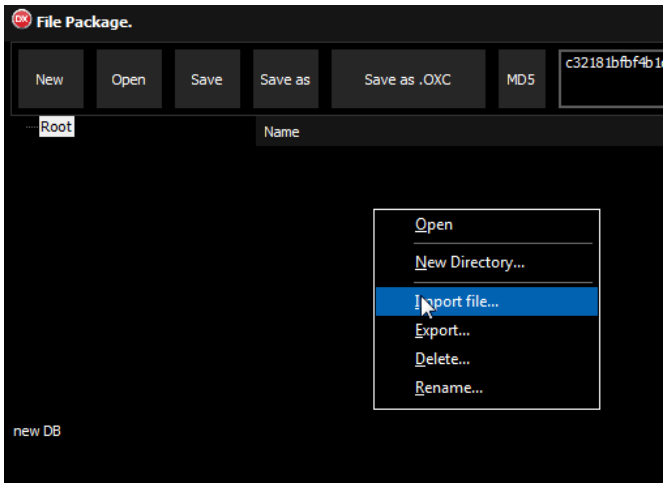
source=输入数据源，支持.ImgDataset+.ImgMat

syncfile=同步文件

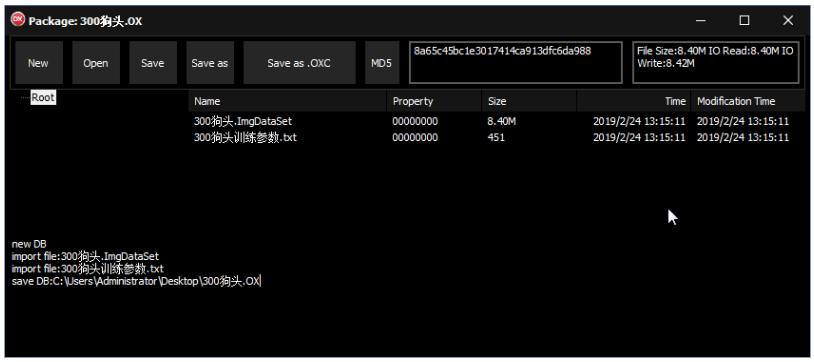
output=输出的模型文件

timeout=训练超时，单位是毫秒，如果写 e”xxx”，xxx 就是 zExpression 表达式规则，zExpression 表达式请参考我的开源项目 https://github.com/PassByYou888/zExpression

在 FilePackageTool 中，将”300 狗头训练参数.txt”+” 300 狗头.ImgData”导入进来



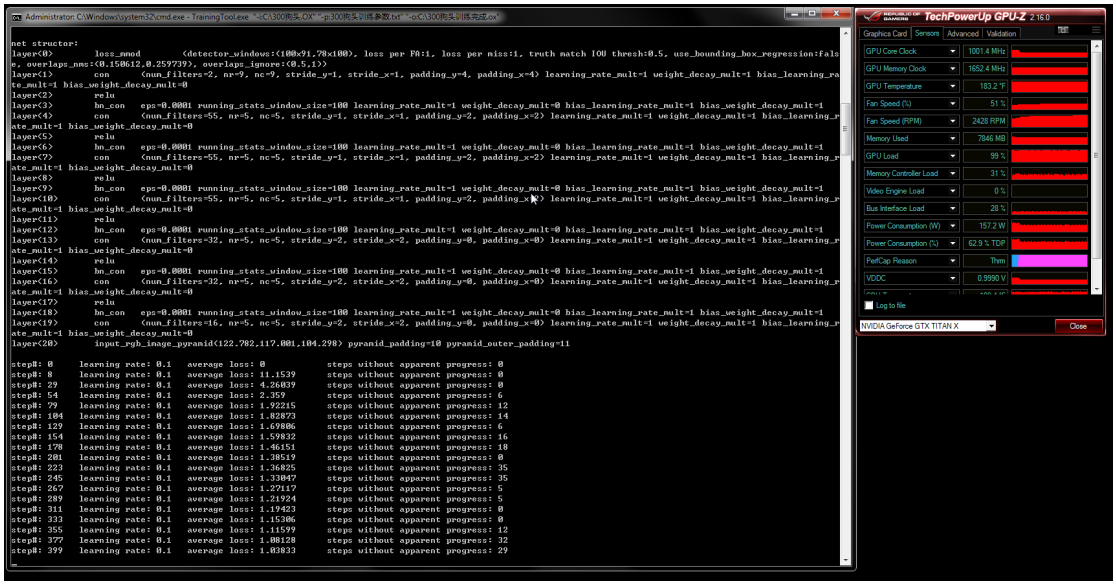
完成后保存成到”300 狗头.ox”，保存到 c:\



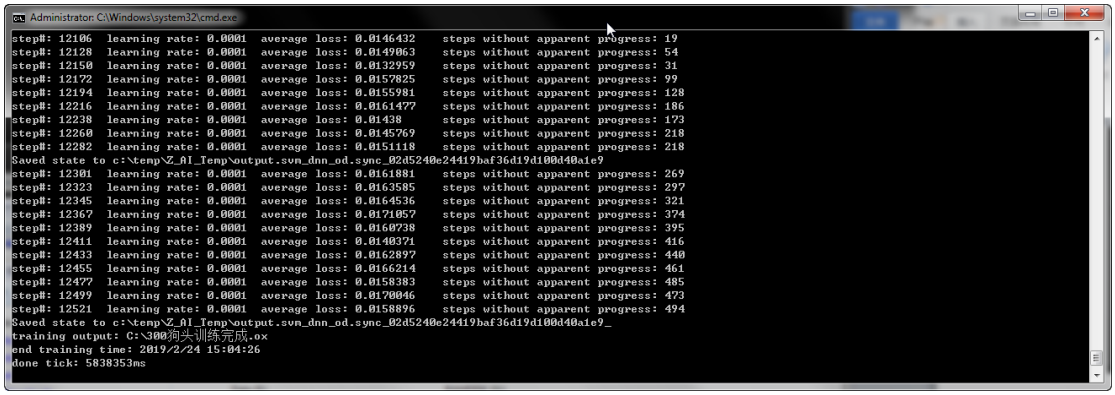
FilePackageTool 在命令行运行

TrainingTool.exe "-i:C:\300 狗头.OX" "-p:300 狗头训练参数.txt" "-o:C:\300 狗头训练完成.ox"

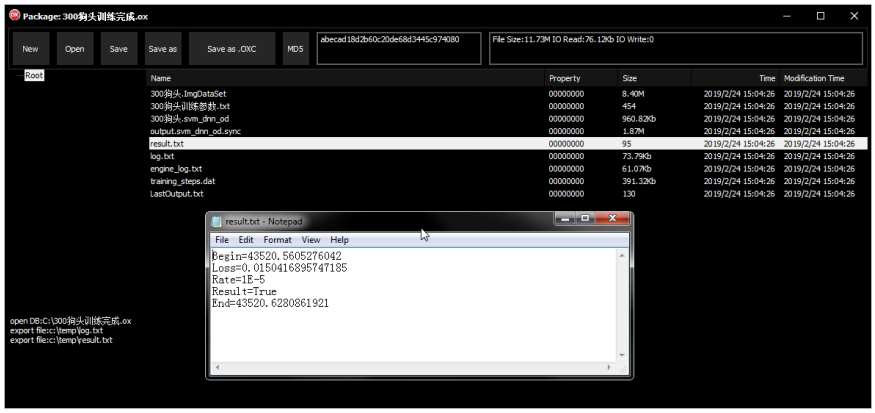
这时候，gpu 已经 99%负荷，显存开销 8G 上下



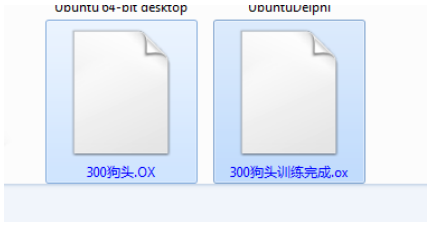
1 小时 50 分后，训练完成.



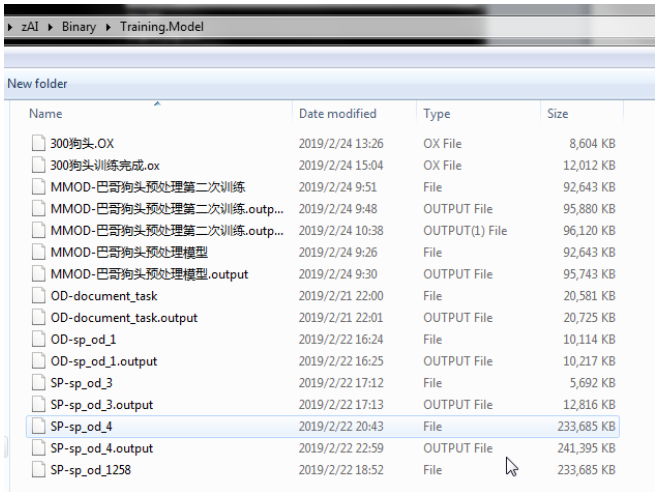
使用 FilePackageTool 打开 C:\300 狗头训练完成.ox，查看 Result.txt，拟合精度在 98.5，这个模型是可以用的



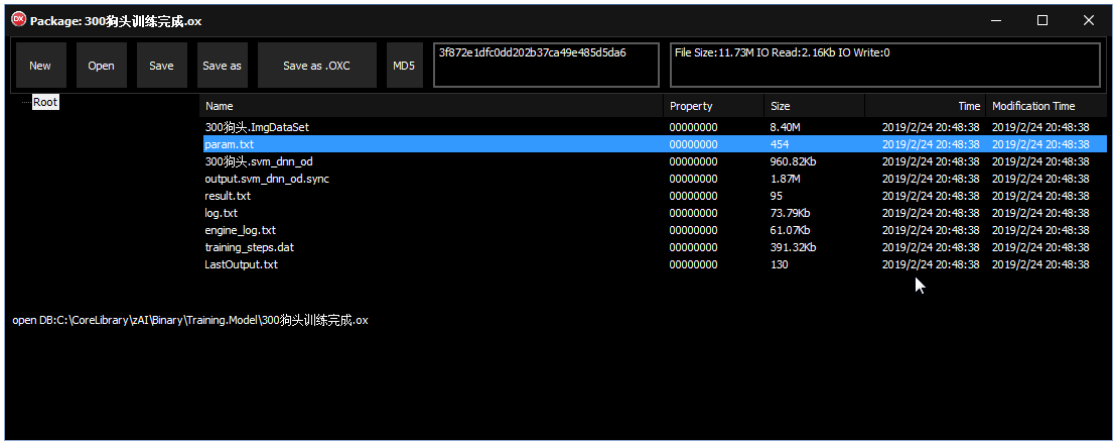
现在，我们可以将 300 狗头 放到服务器去



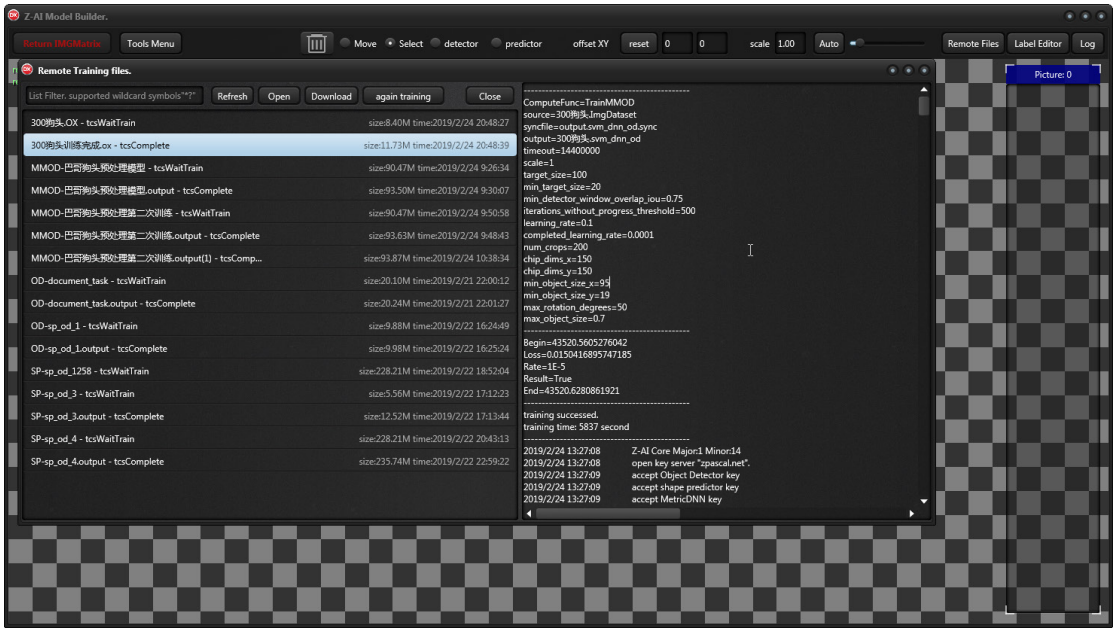
但是，服务器会打不开，因为找不到 param.txt，我们需要修改一下参数文件名



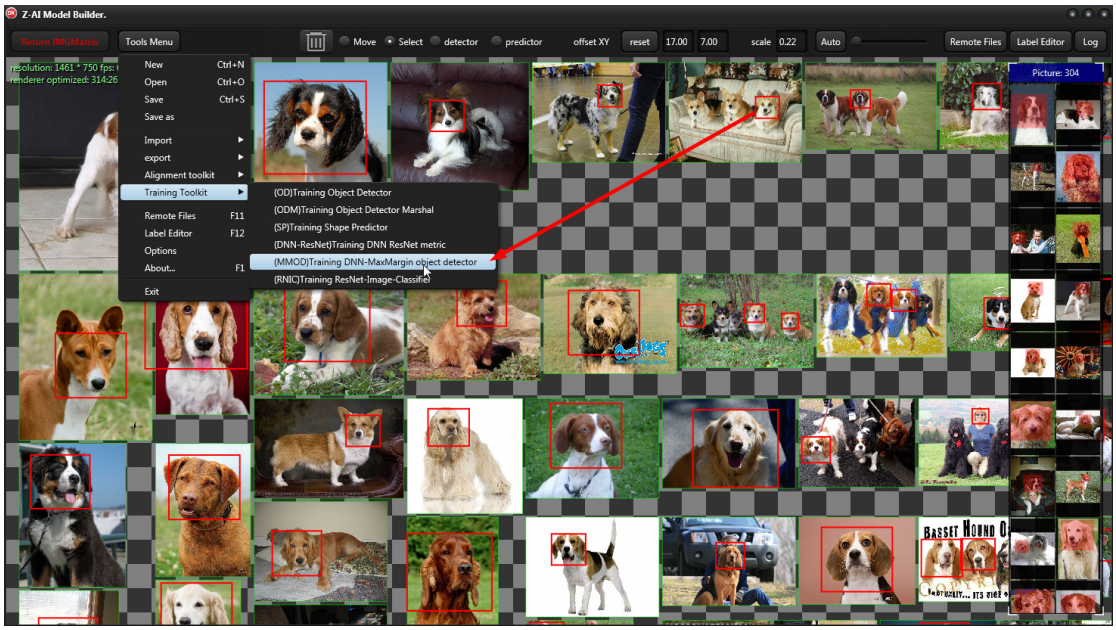
将”300 狗头训练参数.txt”改成”param.txt”



然后在 z_ai_model 中刷新即可，从列表可以看出，我们自定义的 300 狗头.ox 很小，只有 10M 左右

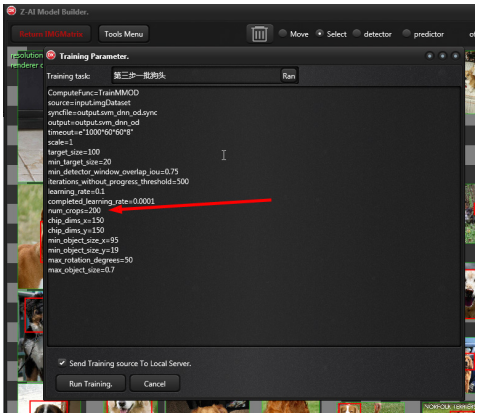


高配设备：暴力的把 300 狗头交给服务器训练

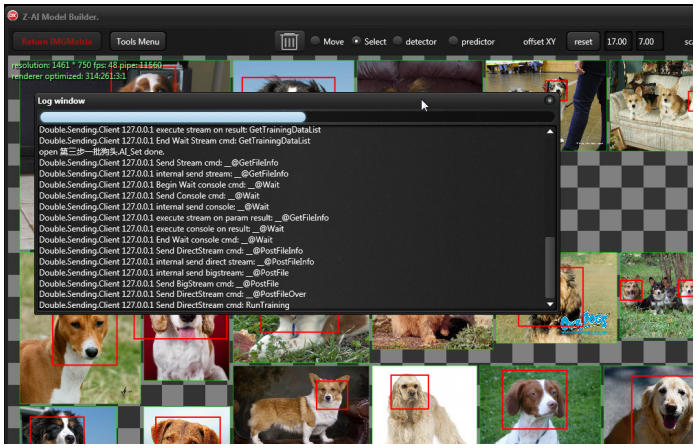


我在这里只改过一个参数，num_crops=200，每一次学习步数，会生成 200 张 resnet 的 image，200 参数会吃掉接近 10G 的显存，这已经接近我的设备配置极限了

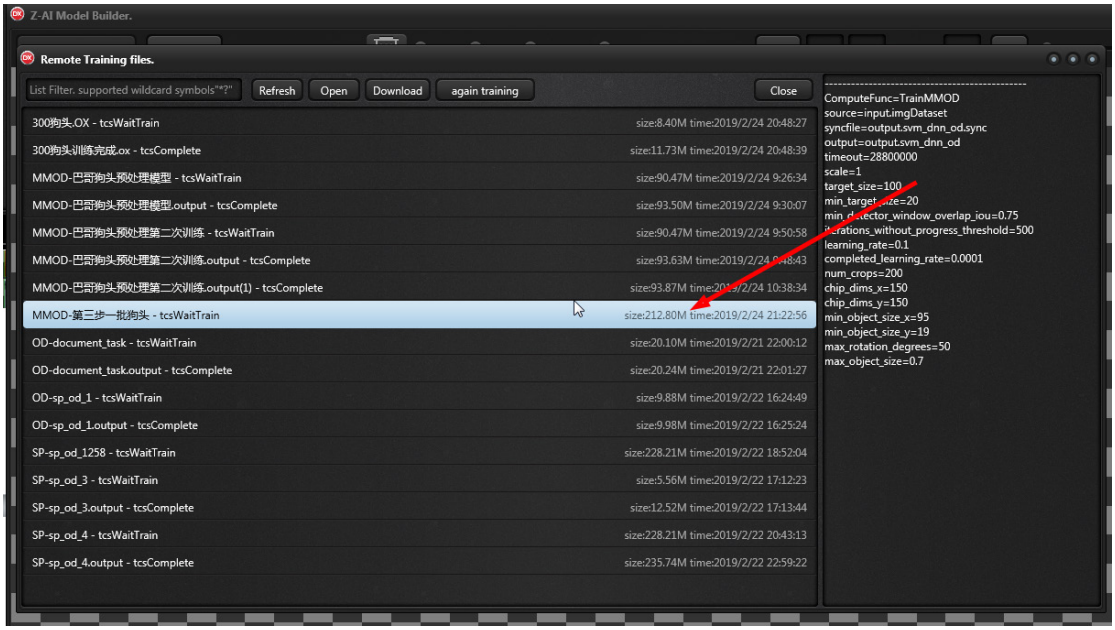
写完以后，发送给服务器训练



这时候，会将训练的数据源发送给服务器，如果带宽不够，发送过程也许会比较长，在数据源发送过程中不要关闭

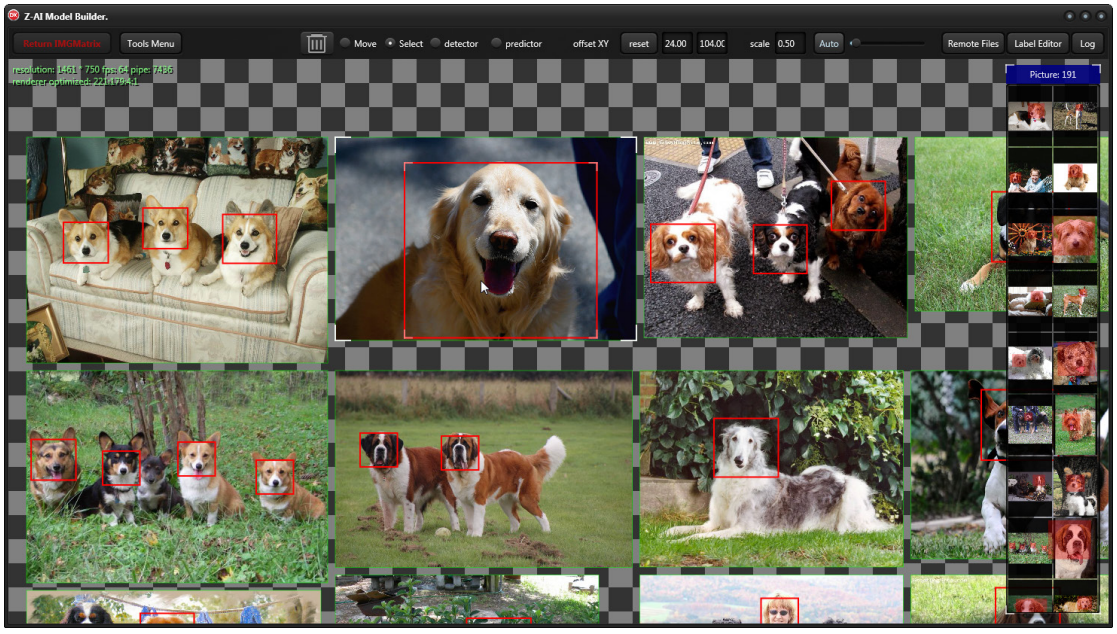


在训练中，我们通过 remote files 可以看道，当前训练的数据文件高达 200M

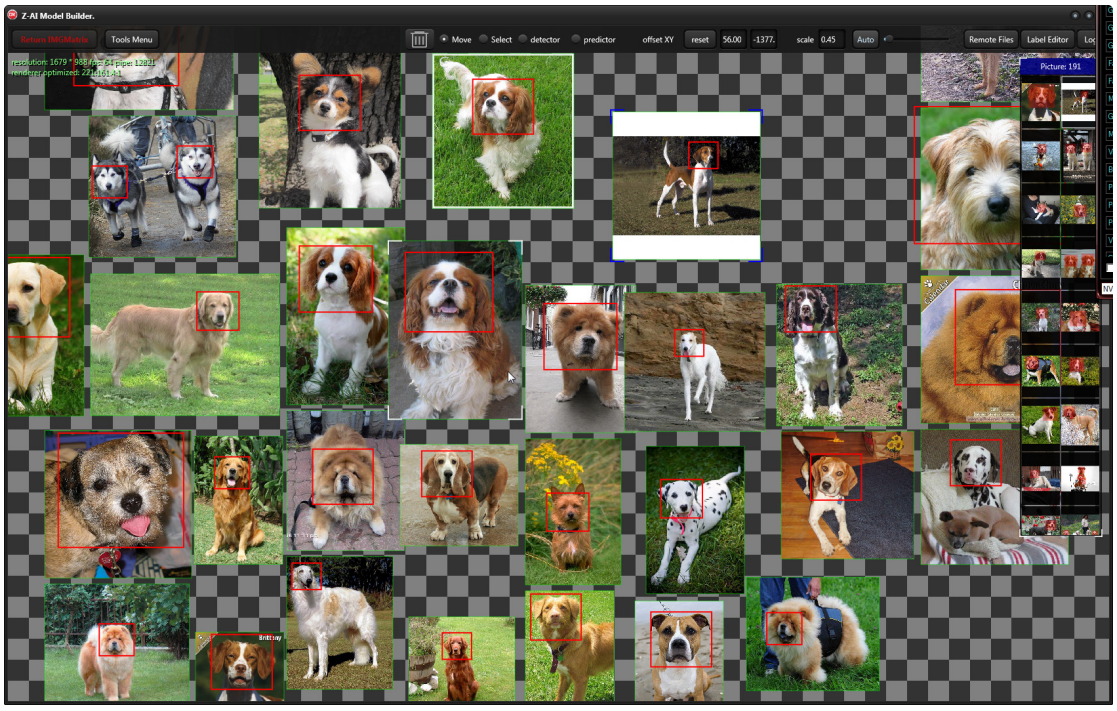


测试训练结果

导入一批小狗，使用我们训练的模型来生成框体测试



很多未曾在我们狗头数据集出现的小狗，都被识别出来了，但也有漏网之鱼

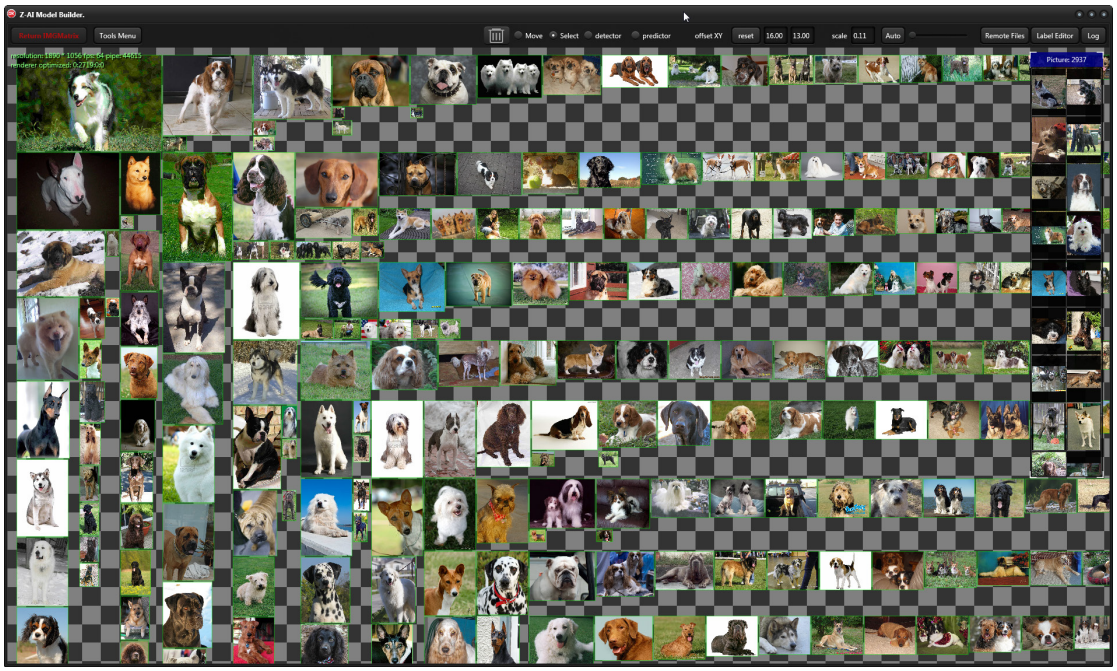


如果测试模型不合格，合并模型，再次训练

从新导入一大批小狗，这里是 3000 张数码小狗图片，然后使用 300 狗头的模型生成框体，检查框体，再次训练，再次测试，依此反复。

这一过程会消耗大量的显存+内存，所以 32G 内存+10G 显存只能算做入门配置，到真正实用，ZAI 对内存+显存的需求还需要我们进一步升级设备。

当然，我们也可以通过细致的 ps 批处理我们的训练数据集，请参考文档
使用 PS 自动脚本批处理.pdf



完
2019-2
By qq600585