Yong Wei
Github

Email : weiyong1024@gmail.com
Mobile : (+86) 186-1263-2313

Education

- **Beihang University** — Beijing, China
  Master of Engineering in Electronic and Communications — *Sep. 2016 – Jan. 2019*
- **Beihang University (Honors College)** — Beijing, China
  Bachelor of Engineering in Electrical and Electronics; GPA: 3.7 — *Sept. 2012 – Jun. 2016*

Experience

- **Pony.ai** — Beijing, China
  Software Engineer - Infrastructure — *Feb 2019 - Mar 2020*
  - **Voice Logging Pipeline**: The original issue recording pipeline needs an onboard engineer to record the log message via keyboard. So it is necessary to provide a voice logging pipeline to enable single-onboard operator. In the new solution, a pair of microphone(for recording) and hand button(for trigger) is equipped for each self-driving vehicle (SDV). First importing the opensource library evdev with BSD license as the input event interface. Then writing a daemon processing based on evdev and ALSA driver from Linux to check the hardware, who listens to signal from the hand button and FIFO connected with the onboard main process, triggers or stops the microphone, and saves the audio files during trips. For the data processing part, first building the speech-to-text service on Google Cloud Platform. Then extract the content from audio files into text. Finally serializing the meta-info by Google Protobuf and create issues by current issue-reporting pipeline.
  - **Car Sound Workflow**: The original Car Sound Module uses Pico library by Google to convert the hardcoded content into audio-stream and then play it out, which is a waste of computing resources and is limited to single Language support. The new car sound module is built on a file-based pipeline, aiming to break those limitations. First, build a internal Python command-line tool (CLI) for engineers to add/update and manage sound suite version: First, write CloudFormation template on AWS to build the Text-to-Speech service, and add a related option in the CLI. Then, add uploading/downloading interface to the storage server in the CLI and enabling local cache. For the onboard part, adding sound names into the sound-request schema, load the audio stream into memory when initializing Car Sound Module, then play the sound when receiving the request.
- **Airbnb** — Beijing, China
  Software Engineer Intern - Web Full Stack — *Jun 2018 - Sep 2018*
  - **Host Retrospect Page**: Retrospect page for Airbnb host users. Added related endpoints and mobile web pages on Ruby On Rails framework. Discussed with the designers on the contents and formats of web pages.
- **Megvii (Face++)** — Beijing, China
  Research Intern — *Jan 2018 - Jun 2018*
  - **Model Search**: During the first half, after the dimension of the input graph increased by 14%, we aimed to reduce the FLOPs (float operations per second) of the CNN (convolutional neural network) of face recognition module on mobile devices to the same level as before, without damaging the performance of the whole model which is measured by $\frac{1}{10,000}$ passing rate (ROC value when $x = \frac{1}{10,000}$). The problem was solved by adding a bottleneck layer before the Inception-ResNet module to compress channels of the input feature graph by $\frac{1}{2}$. As for the second half, we aimed to improve the performance of our face recognition module referring to state-of-the-art CNN architectures. After experiments on Xception, DenseNet and several other architectures, I proposed a modified version of Google Inception V4 for our production: Instead of using the same atom modules everywhere, different Inception-ResNet modules were used for the shallow, medium and deep part of the CNN. On the other hand, the $N \times N$ kernels of the medium and deep part were replaced by a sequence of $1 \times N$ and $N \times 1$ kernels to improve model capacity. As a result, the $\frac{1}{10,000}$ passing rate was improved by 1% on most benchmarks.
- **Beihang University** — Beijing China
  Research — *Jul 2015 - Mar 2016*
  - **The 5^{th} Generation Mobile Network**: Proposed an channel estimation method based on uplink wireless data and channel sparcity with supervisor, improving upper bound of the wireless system's throughput by 28% according to simulation results.
    Paper published in Journal of Signal Processing (First author). DOI: 10.16798/j.issn.1003-0530.2017.06.002

Competitive programming

- **Google Code Jam Kickstart 2017 Round F**
  Top 5%, rank 108^{th} globally. ***scoreboard, id:WeiYong1024***

Programming Skills

- **Languages**: Use C++, Python most commonly; Have experience in Shell, JavaScript,
- **Technologies**: Have experience in AWS, Google Cloud, Kubernetes, React

魏雍
Github
邮箱: weiyong1024@gmail.com
Mobile : (+86) 186-1263-2313

教育背景

- **北京航空航天大学** 中国，北京
  电子与通信工程硕士 *2016 年 1 月 – 2019 年 1 月*
- **北京航空航天大学 (沈元荣誉学院)** 中国，北京
  电子信息工程硕士；*GPA: 3.7* *2012 年 9 月 – 2016 年 6 月*

工作经历

- **小马智行** 中国，北京
  软件工程师 - 基础架构 *2019 年 2 月 - 2020 年 3 月*
  - **行车录音工具链**: 原有 issue 记录流程需要一位工程师跟车通过键盘记录问题的描述信息，故需要增加通过语音记录 issue 的流程供司机使用以支持单人单车运营。为此新增麦克风和按钮作为硬件方案，首先在 Bazel 项目中引入基于 BSD 软件许可的输入设备接口库 evdev，并基于该库和 Linux 的 ALSA 音频驱动实现负责硬件检测、监听来自外部按钮的信号和车载系统进程的管道信息、触发与停止麦克风录音的守护进程 sound_recorder，在行车过程中将 issue 信息以音频文件的形式存储下来。在数据处理阶段，首先在 Google Cloud Platform 上搭建语音转文字服务，在本地使用 Python 脚本将音频文件的内容提取出来，然后使用 Google 的 Protobuf 工具将原信息序列化。最后使用已有的 Issue 汇报系统流程将相关问题发送给 QA 同事。
  - **车载语音系统**: 原有的车载语音模块使用 Google 的 Pico 文字转语音工具库在行车过程中将硬编码的语音内容文本实时播放，从而导致文字转语音的过程需要耗费车载计算资源，同时语言被限制只能使用英文。新的车载语音系统使用基于音频文件的工作流程，旨在减少计算量并支持语音的 I18N。为此首先用 Python 编写内部 click 命令行工具 car_sound_utils(以下简称 CLI)，供工程师用于添加和升级现有语料库、管理语音包版本：首先，在 AWS 上编写 CloudFormation 模板搭建语音转文字的 Web 服务，并在 CLI 中添加对应的调用命令和相应参数，用以在本地生成多语言语音语料。然后，在 CLI 中添加上传、下载语音文件和上传语音包到内部 storage 服务器的接口，并加入本地缓存机制加速下载。对车载系统部分，在播放语音请求的消息原型中加入语音名称，初始化语音模块时使用 Linux 的 ALSA 驱动将音频流加载进内存，并在语音模块接到播放请求消息时播放。
- **爱彼迎** 中国，北京
  软件工程师实习生 - *Web* 全栈 *2018 年 6 月 - 2018 年 9 月*
  - **房东回顾页面**: 开发房东回顾页面。在 Ruby On Rails 框架下实现相关的 Endpoints 和移动段前段页面，与设计师讨论确定前端页面的样式和内容。
- **旷视科技** 中国，北京
  炼丹实习生 *2018 年 1 月 - 2018 年 6 月*
  - **模型搜索**: 在实习期前半段，由于用输入网络的图片空间维度增加了 14%，我们希望能够在不牺牲模型性能的前提下，通过模型压缩将的 CNN(卷积神经网络) 的 FLOPs(每秒钟浮点运算量) 降低到先前的水平。这里模型性能使用万一通过率（ROC 曲线上 $\frac{1}{10,000}$ 时纵坐标的值）来衡量。最终通过对网络中 Inception-ResNet 模块加入 bottleneck 层将输入特征图层数压缩一半的方式使该问题得以解决。在实习期后半段，我们希望借鉴最新的 CNN 结构研究成果进一步提升用于移动端人脸识别模块的性能。在对一些经典网络结构如 Xception、DenseNet 进行测试以后，我提出了一种基于 Google Inception V4 的网络结构用于产品：首先, 先前网络中全部使用相同的 Inception-ResNet 模块，而我针对网络中浅层、中层、深层使用不同结构的模块。另一方面，我通过将 $N \times N$ 卷积核替换为 $1 \times N$ 与 $N \times 1$ 卷积核串联的方式增加了网络容量。最终实验结果显示，上述网络结构使得模型性能在大多数数据集上提升了一个百分点。
- **北京航空航天大学** 中国，北京
  硕士研究生 *2015 年 1 月 - 2016 年 5 月*
  - **第五代移动通信网络**: 在导师指导下提出一种利用上行数据及信道稀疏特性提升信道估计质量的方法，使得网络吞吐量提升了 28%。
    以第一作者合作发表于中文核心期刊《信号处理》。DOI: 10.16798/j.issn.1003-0530.2017.06.002

Coding 能力

- **Google Code Jam Kickstart 2017 Round F**
  前 5%, 全球排名 108^{th}。**计分板链接（*id:WeiYong1024*）**

技术栈

- **编程语言**: C++, Python 最常用；使用过 Shell, JavaScript
- **工具**：使用过 AWS, Google Cloud, Kubernetes, React