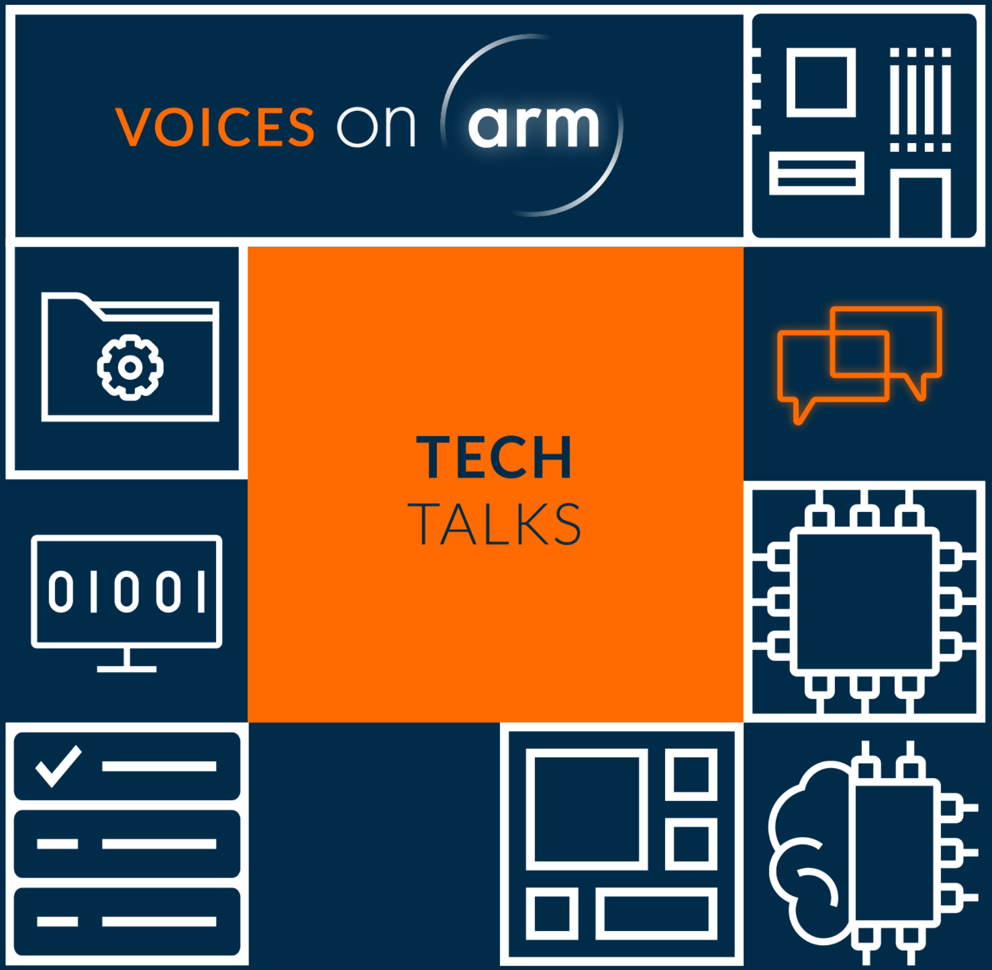
arm

How to Deploy PP-OCR model on Arm Cortex-M with Arm Virtual Hardware



Kai Wang, Liliya Wu
Jan 17th , 2023

Welcome!

Tweet us: #ArmTechTalks

View tech talks on-demand:
www.youtube.com/arm

Sign up for upcoming tech talks:
www.arm.com/techtalks

Our upcoming Arm Tech Talks

Date	Title	Host
January 17th	How to Deploy PP-OCR model on Arm Cortex-M with Arm Virtual Hardware	Baidu
January 24th	Getting Started with Matter with SparkFun and Silicon Labs	Sparkfun & Silicon Labs
January 31st	Bringing Streaming Analytics to Arm-based Edge Devices	Stream Analyze
February 7th	Build Home Automation Services on a Matter Compliant Smart Home Hub Using Python	Arm & Canonical

arm

Kai Wang

Senior product manager with Baidu-PaddlePaddle, with a focus on the technical collaboration with AI hardware companies. Kai collaborates with AI hardware partner such as Arm, NVIDIA and drives the work on PaddlePaddle Ecosystem Distributions, e.g., PaddlePaddle Examples for Arm Virtual Hardware. Prior to joining Baidu, Kai worked for China Mobile and CGG (Houston, USA). He holds a BSc from Peking University and an MSc degree from The University of Texas at Austin.

Kai Wang – Baidu
Senior Product Manager

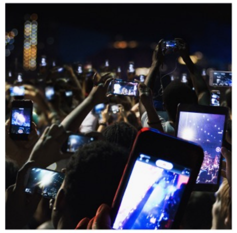
Liliya Wu

Liliya Wu is responsible for supporting the Arm software strategy in the AIoT field, as well as carrying out technical evangelism work around Arm-based solutions and software tool chains. Liliya collaborates with ecosystem partners such as Baidu and Alibaba and promotes the implementation of Arm-based edge AI technology cooperation. She is committed to driving satisfaction with the existing arm tools and platform technologies, improving developer experience and assisting the larger developer community to better understand how to successfully use Arm tools and platform technologies.

Liliya Wu - Arm
Software Engineer – Ecosystem Specialist

arm

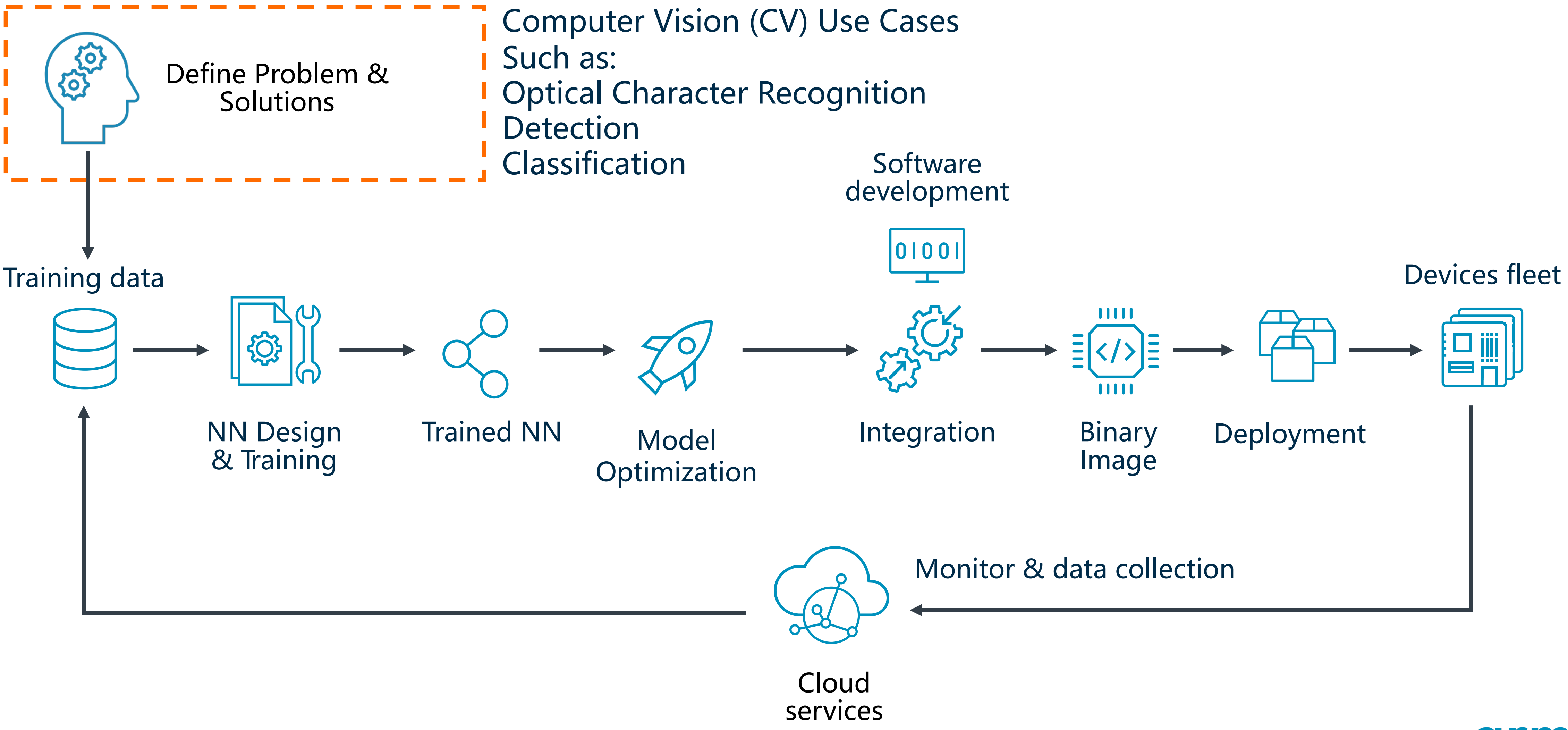
VOICES on arm



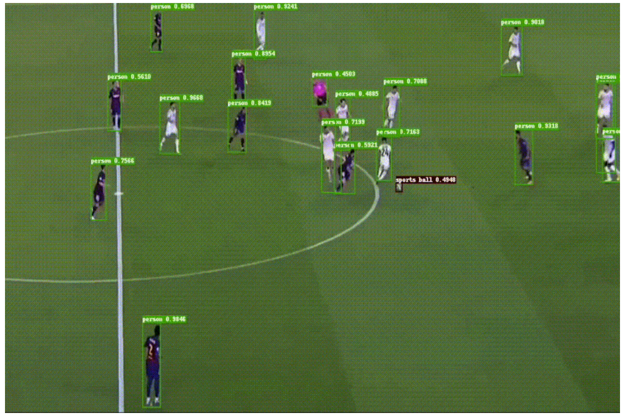
Overview

TECH TALKS

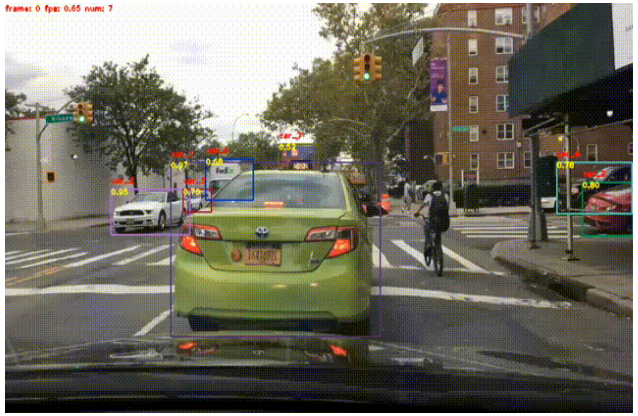
Workflow of a Deep Learning Project



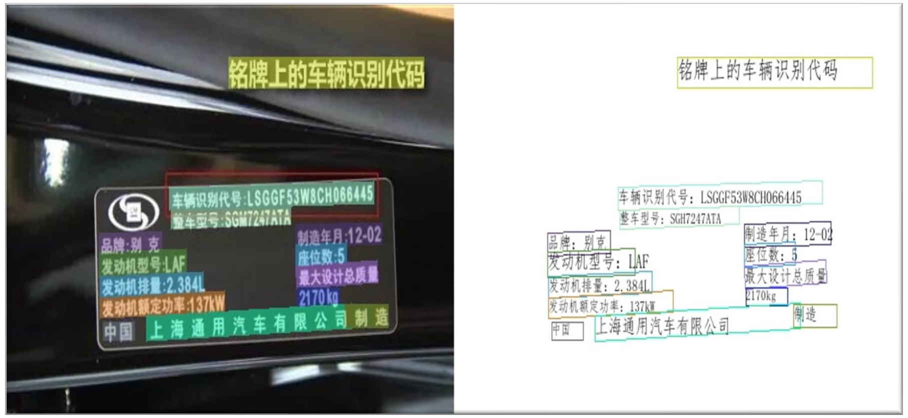
Typical Computer Vision Tasks



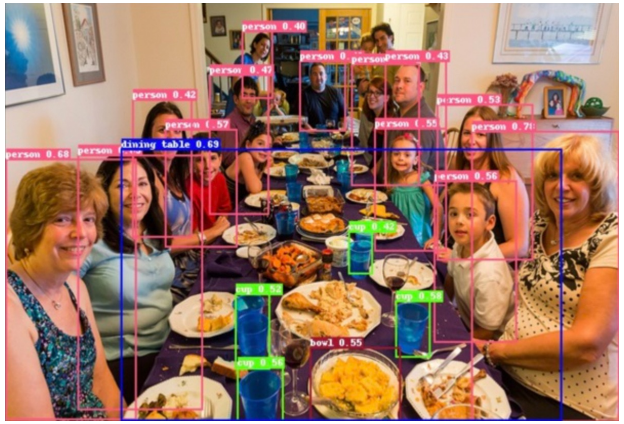
High precision real-time human detection



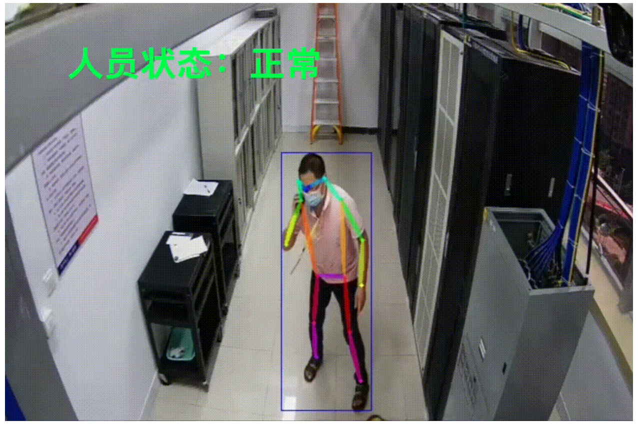
End side real-time vehicle tracking



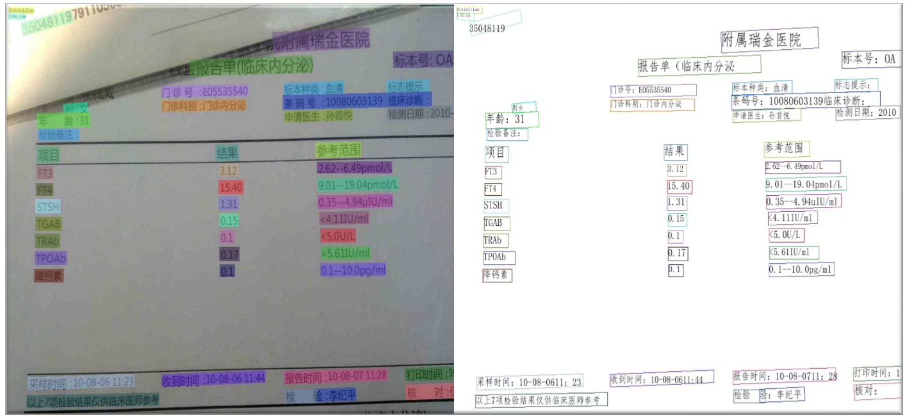
Nameplate identification



Intensive face detection



Fall detection



Identification of test sheet

Object detection and its extended application

Optical Character Recognition （OCR）

Challenges of Deep Learning Application Development

Multiple hardware chips

Multiple algorithm models

TSN
VGG
YOLOv3
ResNet
ICNET
GRU4Rec
UNet
Senta
LAC
SimNet
XLNet
DeepCTR
ERNIE
BERT
MobileNet
DeepSpeech2
Transformer
DQN
DeepLabv3+

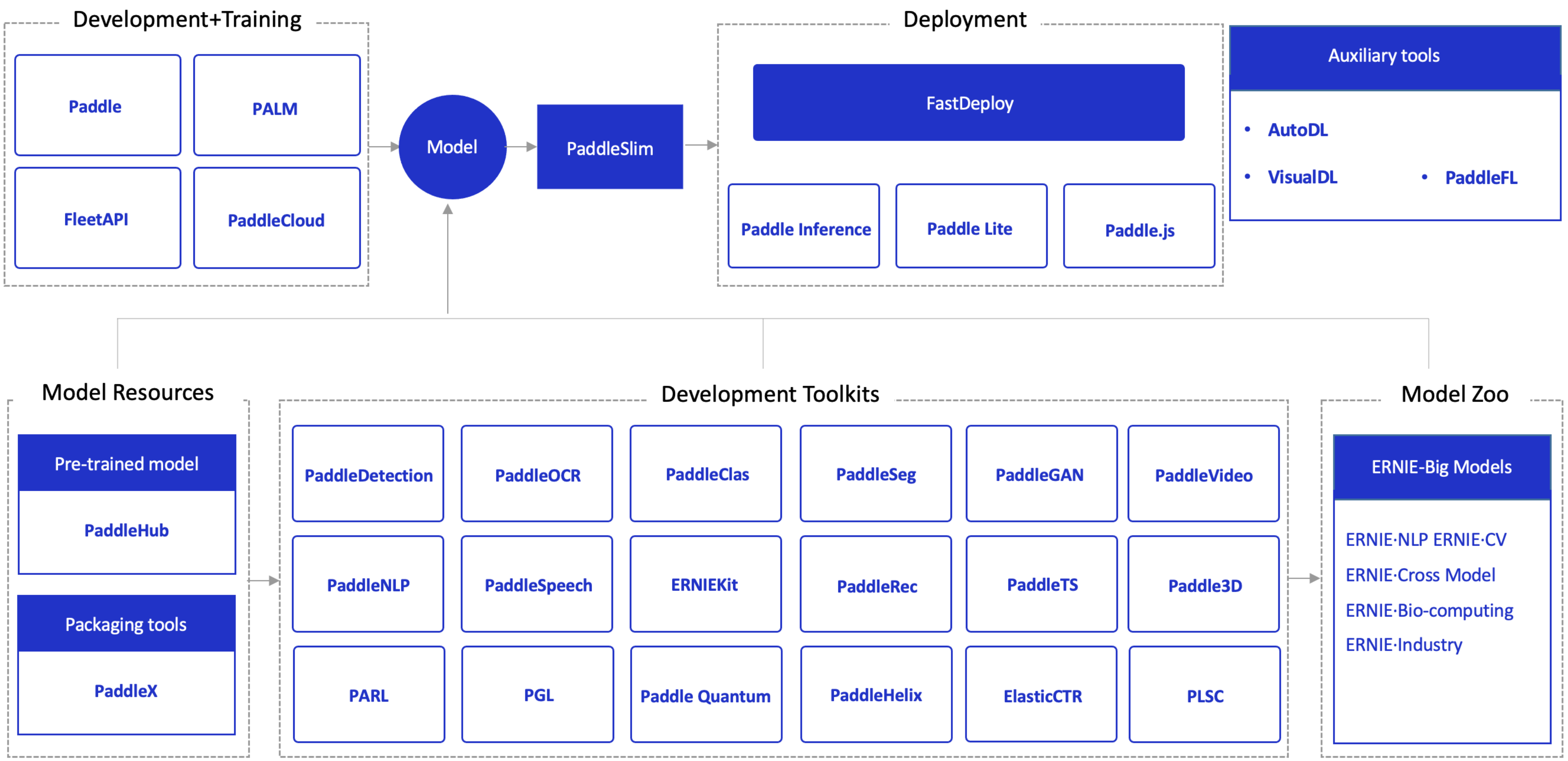
Multiple application scenarios

Complex adaptation and deployment

Difficulties in application development

arm

Overview of PaddlePaddle Open Source System



Core Technologies of PaddlePaddle

Conveniently developed deep learning framework

- The first dynamic and static unified framework in the industry
- Dynamic diagram programming debugging to static diagram prediction deployment

Training technology of large-scale deep learning model

- The first general heterogeneous parameter server architecture in the industry
- End to end adaptive distributed training architecture

High Performance Inference Engine **Deployed** on Multiple Terminals and Platforms

- Ready to use
- Support multi hardware and multi operating systems of end cloud

Industry level open-source **Model Zoo**

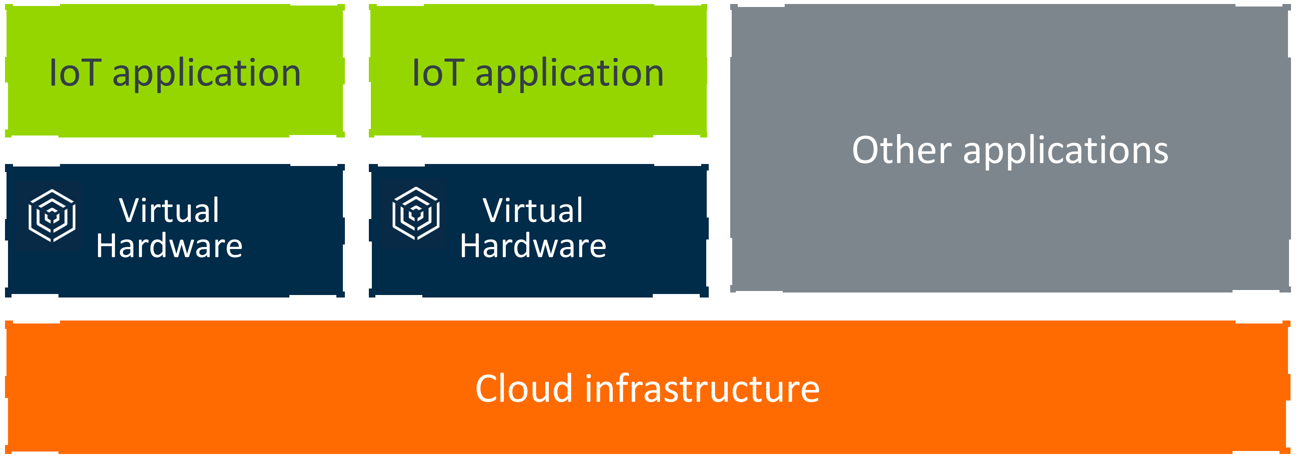
- The total number of algorithms exceeds 600
- Including leading pre-trained model

arm

Arm Virtual Hardware (AVH)

What is arm Virtual Hardware ?

- Virtual, functional representation of a physical hardware
- Cloud-native - runs and scales easily in the cloud
- Suitable for all IoT workloads from MCUs through to Intelligent Edges
- No dependency on RTL or silicon availability



Arm Virtual Hardware (AVH)

Integration of PaddlePaddle and Arm Virtual Hardware sparks an efficient MLOps

- **Cloud development**

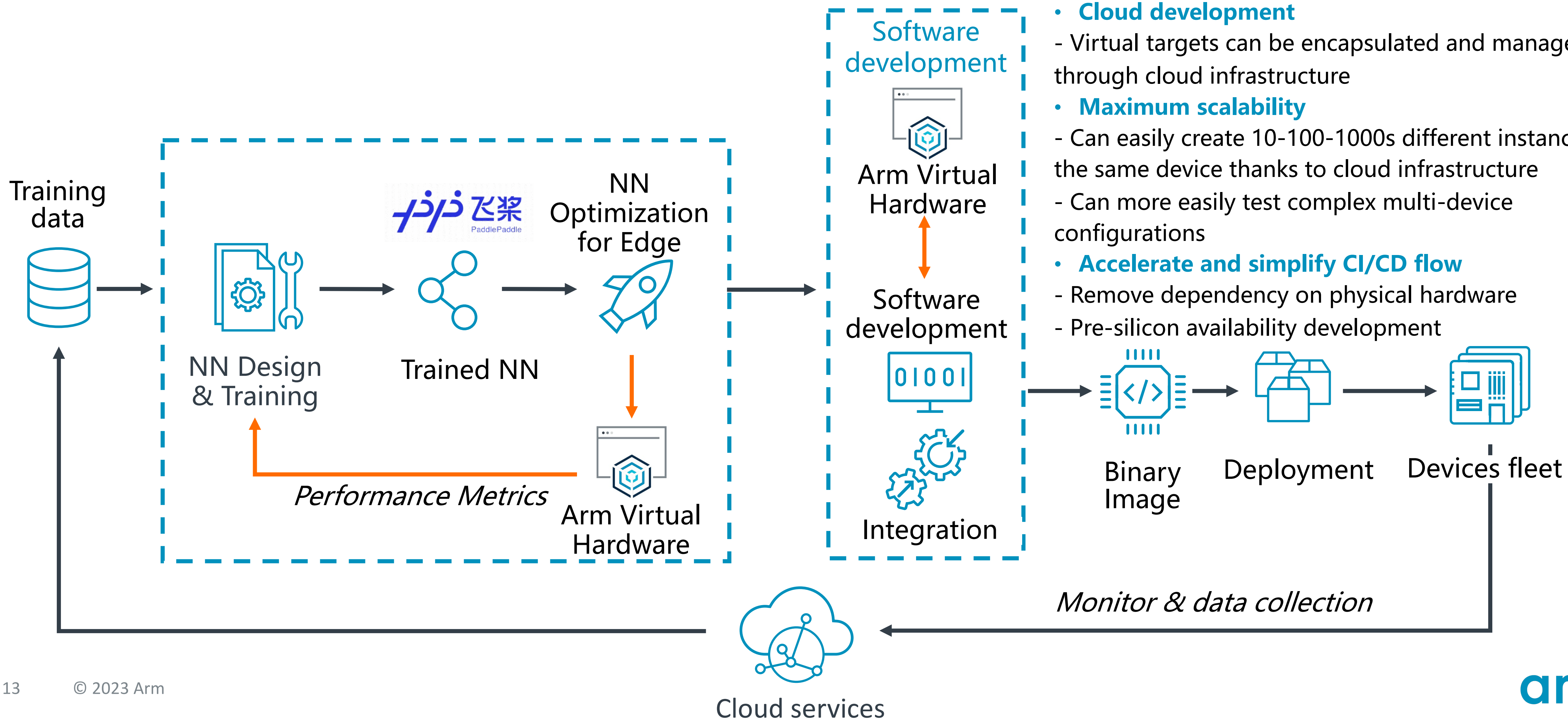
  - Virtual targets can be encapsulated and managed through cloud infrastructure
- **Maximum scalability**

  - Can easily create 10-100-1000s different instances of the same device thanks to cloud infrastructure

  - Can more easily test complex multi-device configurations
- **Accelerate and simplify CI/CD flow**

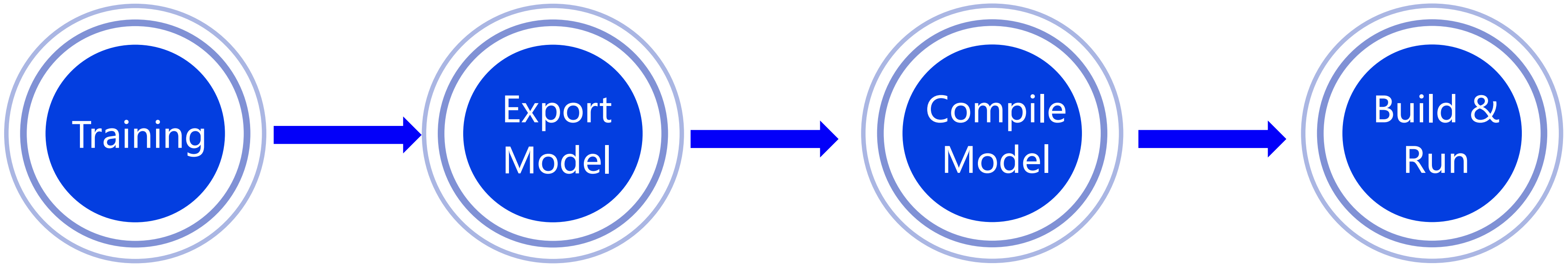
  - Remove dependency on physical hardware

  - Pre-silicon availability development



What will you learn?

An end-to-end workflow to deploy PaddlePaddle model on Arm Virtual Hardware



Obtain trained model

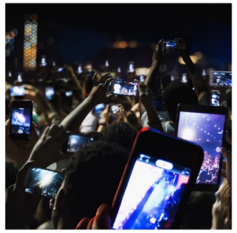
Export trained model to Paddle Inference model

Compile Paddle Inference model to target machine code using TVM

Build application and run in Arm® Cortex®-M55 using Arm Virtual Hardware

- How to prepare a PaddlePaddle trained model with PaddleOCR(GitHub) dev kit
- How to export the PaddlePaddle trained model to PaddlePaddle inference model
- How to use TVMC to compile PaddlePaddle inference model for target device
- How to build the application and deploy/test it on Arm Cortex-M55 using AVH

PaddleOCR Overview



VOICES on arm

TECH
TALKS

arm

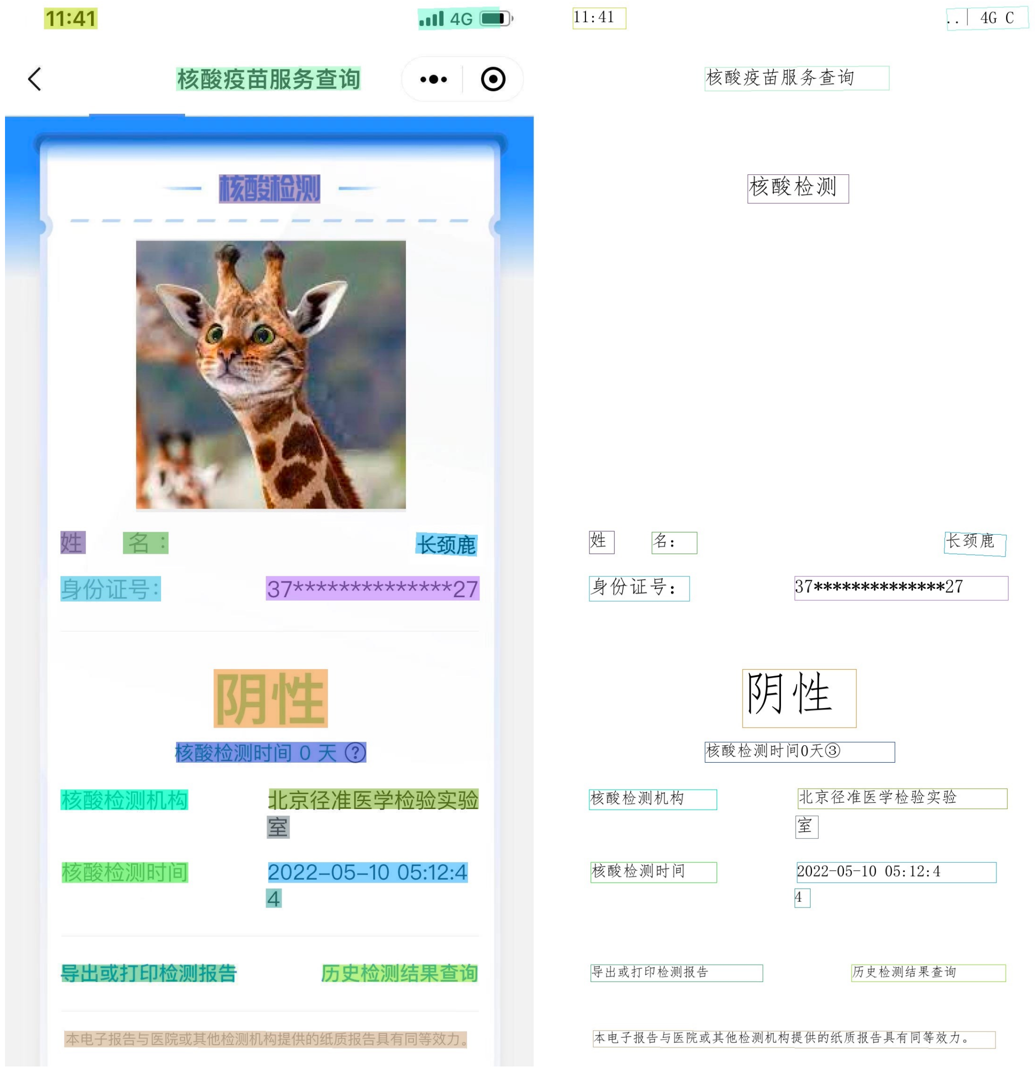
Content

飞桨 PaddleOCR

What is OCR?

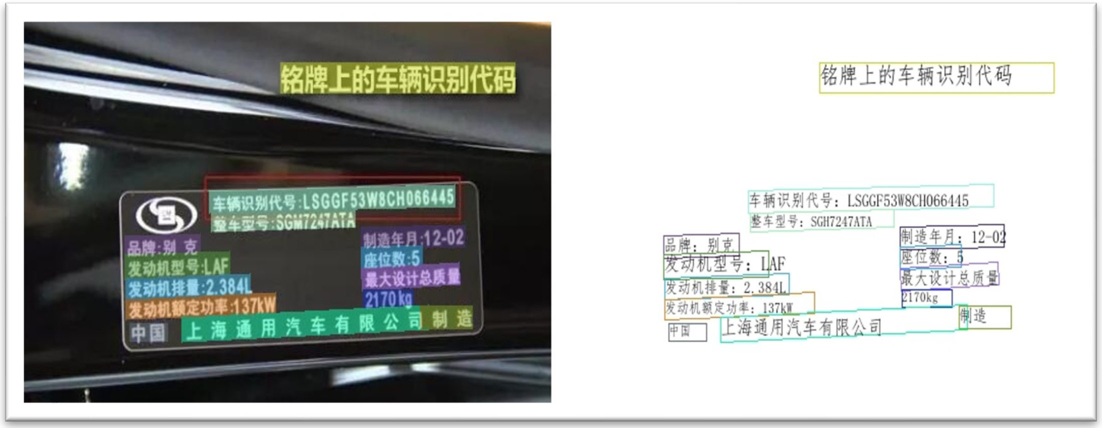
OCR means *Optical character recognition*

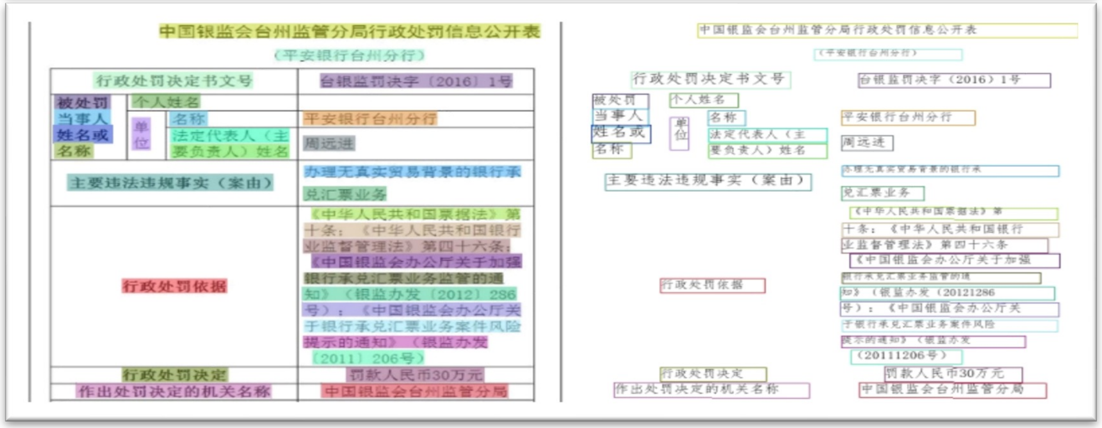
- Extract text from images.
- Free human beings from repetitive work.

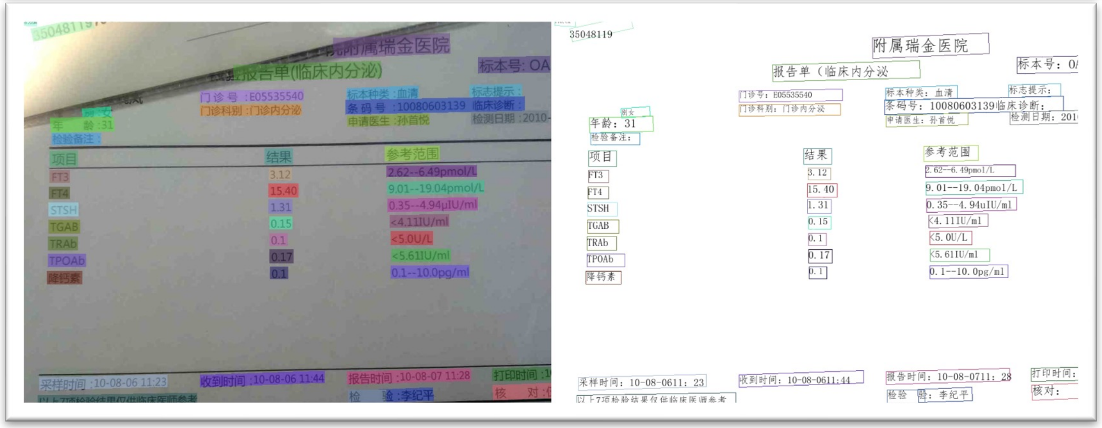


Typical Industrial Application Scenarios of OCR

Information extraction, entry and review of card, certificate and bill, factory automation, electronization of hospital and other documents, online education, etc







Typical Industrial Application Scenarios of OCR

- Video scene (text in the video: subtitles, titles, advertisements, bullet screens, etc.)
  - Text based content: subtitle translation, content security monitoring, etc.
  - Combined with visual features: video understanding, video retrieval, etc.

[Image Source: The picture is from the network]

1. 六个核桃
2. 最强大脑
3. 第二轮规则
4. 第二轮,人机共同观察一位30岁以上的观众
5. 随后将他从30张小学毕业照中找出,正确得
6. 2分
7. 是决定性的一局

Video understanding
Video retrieval
Content monitoring
subtitle translation
…

Challenges of the Implementation of the OCR Industry

Can't find
Can't select

- Requirement: no open-source project can be found
- Item: no key algorithm can be selected

Not applicable to industrial scenarios

- Academic models mainly pursue model accuracy
- Lack of prediction time limits in practice

Difficult to optimize too many problems

- High cost of verifying various optimization methods
- Frequently go astray from training to deployment

Difficulty in data acquisition and annotation

- High data acquisition cost for specific scenarios
- Intense presence of text in images

arm

arm

Content

- Background Overview
  - What is OCR?
  - Typical industrial OCR application scenarios.
  - Challenges of OCR application.
- PaddleOCR Development Kit
  - PaddleOCR Overview
  - PP-OCRv3
- Model Adaptation and Transfer
  - Network adaptation
  - Model training
  - Model export
  - Pre and post processing

飞桨 PaddleOCR

PaddleOCR Panorama – Overview

Application

Financial scene
- Forms
- Bills

Industrial scene
- Watt hour meter
- License plate

Educational scene
- Handwriting
- Formula

Medical scene
- laboratory test report

Deployment

Training mode
- Normal
- Distributed
- Mixed Precision

Training Env.
- Linux GPU/CPU
- Linux DCU
- Windows GPU/CPU
- macOS

Compression
- Pruning
- Quantization
- Distillation

Inference and Deployment
- Python/C++ Inference
- Python/C++ Serving
- OpenCL ARM GPU
- Paddle2ONNX
- PaddleCloud
- Arm CPU
- Jetson
- Paddle.js

Industrial models and solutions

PP-OCR: Ultra-lightweight OCR System 🔥
- PP-OCRv3: detection + direction classifier + recognition =17.0M
- English & numbers model: applicable to scenarios which only contain English and numbers.
- Multilingual models: support 80 languages including Korean, Japanese, German, French, etc.

PP-Structure: Structured Document Analysis System 🔥
- Support layout analysis
- Support table recognition (support export to Excel)
- Support KIE (including semantic entity recognition & relation extraction)
- Support layout recovery
- Support PDF to Word

Algorithms

Text Detection
- EAST
- DB
- SAST
- PSENet
- FCENet
- ……

Text Recognition
- CRNN
- SRN
- NRTR
- SVTR
- ABINet
- ……

End-to-end
- PGNet

Layout Analysis
- Layoutparser
- PP-Picodet

Table Recognition
- TableRec-RARE
- TableMaster
- SLANet

Key Information Extraction
- SDMGR
- LayoutLM
- LayoutLMv2
- LayoutXLM
- VI- LayoutXLM

Community
- Official discussion group
- Contributor Honor Wall
- Regular season challenge

E-book: *Dive into OCR*
- Comprehensive OCR technology
- Theory + Practice
- Notebook interactive learning
- Teaching video

Data tools
- Semi-automatic data annotation tool: PPOCRLabel
- Data synthesis tool: Style-Text

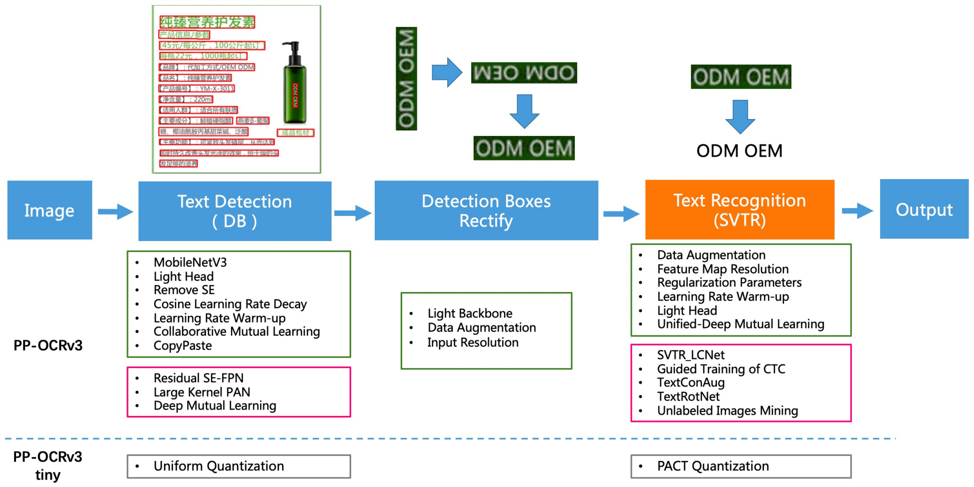
PP-OCRv3 Framework

Text detection

- LK-PAN: PAN structure of large receptive field
- DML: Teacher Model Mutual Learning Strategy
- RSE-FPN: FPN Structure of Residual Attention Mechanism

Text recognition

- SVTR_ LCNet: Lightweight Text Recognition Network
- GTC: Attention guides CTC training strategy
- TextConAug: Data Augmentation Strategy for Mining Text Context Information
- TextRotNet: Self-supervised Pre-trained Model
- UDML: Joint Mutual Learning Strategy
- UIM: unlabeled data mining scheme



Model advantages

- High precision, 5% higher than PP-OCRv2 for Chinese scenes
- Fast inference speed with the single CPU taking 331ms.
- Small model, detection (3.6M)+direction classifier (1.4M)+recognition (12M)=17.0M -> adaptive version recognition model 2.7M

arm

Content

- Background Overview
  - What is OCR?
  - Typical industrial OCR application scenarios.
  - Challenges of OCR application.
- PaddleOCR Development Kit
  - PaddleOCR Overview
  - PP-OCRv3
- Model Adaptation and Transfer
  - Network adaptation
  - Model training
  - Model export
  - Pre and post processing

飞桨 PaddleOCR

Model Adaptation

Configure to remove unsupported operators and retrain the model with the PaddleOCR kit

Operators and external libraries used in PP-OCRv3 English text recognition model

- Backbone:
  - Conv
  - Fully Connected
  - Pooling
  - Softmax
  - Bn
  - Relu
- Neck:
  - LayerNorm (Not supported)
  - LSTM (Not supported)

* note: Not supported means the operators are not supported as front-end operators for TVM.

Operators supported by CMSIS-NN (updating)

- For more information, please visit https://github.com/ARM-software/CMSIS-NN

Operator	C int8	C int16	DSP int8	DSP int16	MVE int8	MVE int16
Conv2D	Yes	Yes	Yes	Yes	Yes	Yes
DepthwiseConv2D	Yes	Yes	Yes	Yes	Yes	Yes
Fully Connected	Yes	Yes	Yes	Yes	Yes	Yes
Add	Yes	Yes	Yes	Yes	Yes	Yes
Mul	Yes	Yes	Yes	Yes	Yes	Yes
MaxPooling	Yes	Yes	Yes	Yes	Yes	Yes
AvgPooling	Yes	Yes	Yes	Yes	Yes	Yes
Softmax	Yes	Yes	Yes	Yes	No	No
LSTM	No	No	No	No	No	No

Model Adaptation

PaddleOCR uses configuration files to control network training & evaluation parameters.

Original configuration file – GitHub

```yaml
Architecture:
  model_type: rec
  algorithm: SVTR
  Transform:
  Backbone:
    name: MobileNetV1Enhance
    scale: 0.5
    last_conv_stride: [1, 2]
    last_pool_type: avg
  Head:
    name: MultiHead
    head_list:
      - CTCHead:
          Neck:
            name: svtr
            dims: 64
            depth: 2
            hidden_dims: 120
            use_guide: True
          Head:
            fc_decay: 0.00001
      - SARHead:
          enc_dim: 512
          max_text_length: *max_text_length
```

Adjusted configuration file – Blog

```yaml
Architecture:
  model_type: rec
  algorithm: SVTR
  Transform:
  Backbone:
    name: MobileNetV1Enhance
    scale: 0.5
    last_conv_stride: [1, 2]
    last_pool_type: avg
  Neck:
    name: SequenceEncoder
    encoder_type: reshape
  Head:
    name: CTCHead
    mid_channels: 96
    fc_decay: 0.00002
```

Model Optimization

The model optimization part uses BDA (Base Data Augmentation), which includes multiple basic data enhancement methods such as random clipping, random fuzzy, random noise, image color inversion, etc.

Original image

Random clipping

Random fuzzy

Random noise

Image color inversion

Model Training

- Dataset: Online open-source datasets MJSynth and SynthText (MJ+ST)
- Training Command:

```
python tools/train.py -c configs/rec/simple_net.yml -o \
    Global.save_model_dir=output/rec/ \
    Train.dataset.name=LMDBDataSet \
    Train.dataset.data_dir=MJ_ST \
    Eval.dataset.name=LMDBDataSet \
    Eval.dataset.data_dir=EN_eval
```

  - Note: modify the dataset location and model file save location respectively based on your need.
- Help Guide – Refer to the documents for more information on model training with PaddleOCR development kit.

arm

Model Export and Function Verfication

Export PaddlePaddle inference model

- We must export the trained text recognition model to a Paddle inference model that we can compile to generate code which is suitable to run on a Cortex-M processor.
- Use the following command to export the Paddle inference model. We need manually modify the output shape to [3,32,320].

```
python tools/export_model.py -c configs/det/ch_PP-OCRv3/ch_PP-OCRv3_det_student.yml -o \
    Global.pretrained_model=output/rec/best_accuracy.pdparams \
    Global.save_inference_dir=output/rec/infer
```

Verify the model inference effect on PC

- You can use PaddleOCR toolkit to verify the model inference effect. Visit the documents to learn how to verify the results.

```
python3 tools/infer/predict_rec.py \
    --image_dir=./doc/imgs_words/ch/word_4.jpg \
    --rec_model_dir=output/rec/infer
```

Pre and Post Processing

Pre-Processing

- To ensure the picture you use is in right shape, you can refer to the script for pre-processing.

Post-Processing

- C language coding is required according to specific tasks.
- For this example, the model is OCR English recognition model, so only post-processing of the recognition model is included.
- Post recognition processing: for the output of the model, shape is B * W * N, Where W is the maximum recognized character; N is the size of the dictionary; B is the batch. Post-processing is to find each recognized character on W.

```
printf("text: ");
for (int i = 0; i < char_nums; i++) {
  int argmax_idx = 0;
  float max_value = 0.0f;
  for (int j = 0; j < char_dict_nums; j++){
    if (output[i * char_dict_nums + j] > max_value){
      max_value = output[i * char_dict_nums + j];
      argmax_idx = j;
    }
  }
  if (argmax_idx > 0 && (!(i > 0 && argmax_idx == last_index))) {
    score += max_value;
    count += 1;
    // printf("%d, %f, %c\n", argmax_idx, max_value, dict[argmax_idx]);
    printf("%c", dict[argmax_idx]);
  }
  last_index = argmax_idx;
}
score /= count;
printf(", score: %f\n", score);
```

Calculate the characters of each step.

arm

Compile Model with TVMC

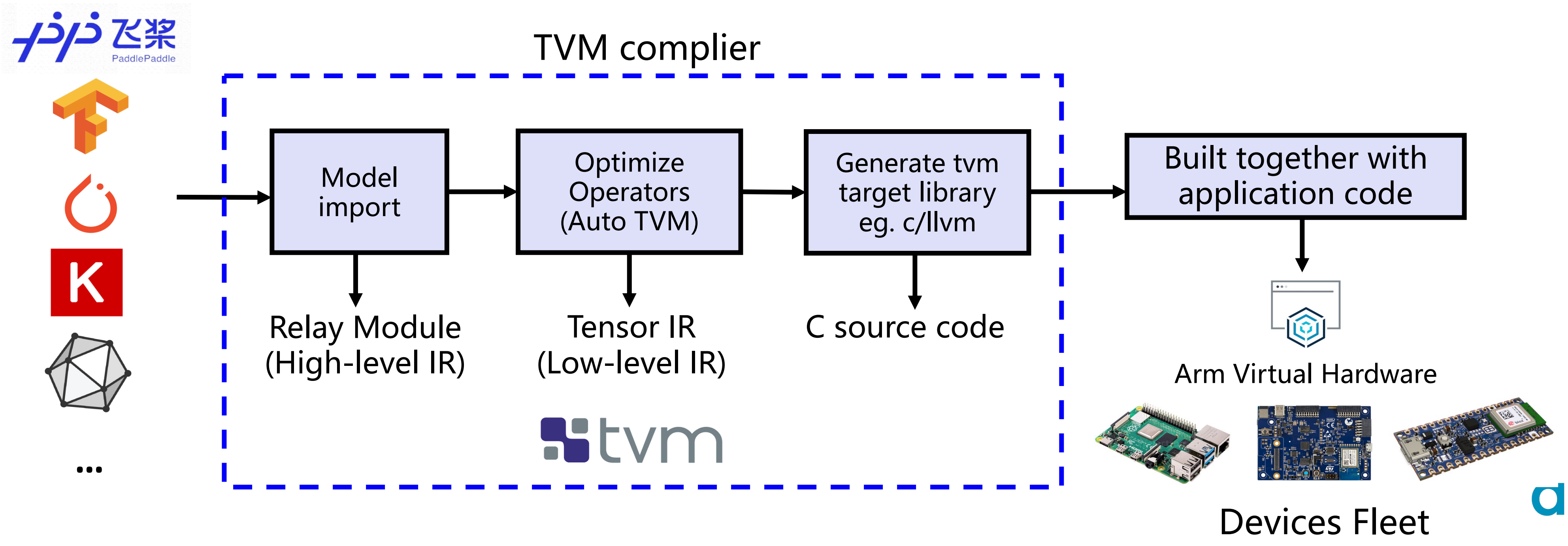
VOICES on arm

TECH TALKS

MicroTVM: TVM on bare-metal

Learn more about Apache TVM at https://tvm.apache.org/

- TVM is an open-source Deep Learning Compiler Stack that closes the gap between the productivity-focused deep learning frameworks, and the performance-oriented or efficiency-oriented hardware backends.
- MicroTVM runs TVM models on bare-metal (such as IoT) devices



PaddlePaddle as TVM Front-end

Refer to the guide to learn more about how to compile PaddlePaddle models using TVM

- PaddlePaddle is officially supported as TVM front-end in TVM v8.0 releasement!
  - Support 120+ operators and 100+ models.
- Future Plan:
  - Support 200+ operators.
  - Support stream operator.
  - Support models quantized by PaddleSlim.

```
import paddle
import paddle.vision.models as models

model = models.renset50(pretrained=True)
model.eval()
# save model as static model
input_spec =
paddle.static.InputSpec(dtype="float32",
       shape=[None, 3, 224, 224], name="image")
paddle.jit.save(model, "save_dir/model",
[input_spec])
```

Export the model

```
import paddle
from tvm import relay

model = paddle.jit.load(“./inference/model”)
mod, params = relay.frontend.from_paddle(model)

with tvm.transform.PassContext(opt_level=3):
       lib = relay.build(mod, target,
params=params)
```

Compile the model with TVM IR (Relay)

Offloading to CMSIS NN

Learn more about CMSIS NN at https://github.com/ARM-software/CMSIS-NN

- CMSIS NN software library is a collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Arm Cortex-M processors.
- Using CMSIS NN with TVM
  - TVM allows for partitioning and code generation using an external compiler.
  - Partitioned subgraphs containing operators targeted to Cortex-M can then be translated into the CMSIS NN C APIs.
- Learn more about CMSIS NN and TVM integration details at GitHub.
  - Supported operators(updating) can be found in the script.

```
TVM_DLL int32_t
  tvmgen_detection_cmsis_nn_main_2(int8_t* input_, int8_t* filter_, int32_t* multiplier_,
                                   int32_t* filter_scale_, int32_t* bias_, int32_t* input_scale_,
                                   int32_t* shift_, int8_t* output_,
                                   uint8_t* global_workspace_3_var) {
  cmsis_nn_context context = {NULL, 0};
  cmsis_nn_tile stride = {1, 1};
  cmsis_nn_tile padding = {0, 0};
  cmsis_nn_tile dilation = {1, 1};
  cmsis_nn_activation activation = {-128, 127};
  cmsis_nn_conv_params conv_params = {128, -128, stride, padding, dilation, activation};
  cmsis_nn_per_channel_quant_params quant_params = {multiplier_, shift_};
  cmsis_nn_dims input_dims = {1, 48, 48, 8};
  cmsis_nn_dims filter_dims = {16, 1, 1, 8};
  cmsis_nn_dims bias_dims = {1, 1, 1, 16};
  cmsis_nn_dims output_dims = {1, 48, 48, 16};
  arm_status status =
      arm_convolve_wrapper_s8(&context, &conv_params, &quant_params, &input_dims, input_,
                              &filter_dims, filter_, &bias_dims, bias_, &output_dims, output_);
  if (status != ARM_MATH_SUCCESS) {
    return -1;
  }
  return 0;
}
```

TVMC – TVM Command Line Driver

- TVMC is a Python application. When you install TVM using a Python package, you will get TVMC as a command line application called tvmc that exposes TVM features such as auto-tuning, compiling, profiling and execution of models through a command line interface.
- Example – The command will:
  - Offload operators to CMSIS-NN, falling back to C code.
  - Set target device to Cortex-M55.
  - Use aot (Ahead Of Time compiltaion) as executor to compile the model.
  - The model file path is under ocr_en/inference.pdmodel
  - The model is in PaddlePaddle format
  - The output format is Model Library Format (only for microTVM targets)
  - The output package will be named as rec.tar under current directory
  - For more descriptions of each parameter, use `tvmc compile --help` to check.

```
python3 -m tvm.driver.tvmc compile --target=cmsis-nn,c \
    --target-cmsis-nn-mcpu=cortex-m55 \
    --target-c-mcpu=cortex-m55 \
    --runtime=crt \
    --executor=aot \
    --executor-aot-interface-api=c \
    --executor-aot-unpacked-api=1 \
    --pass-config tir.usmp.enable=1 \
    --pass-config tir.usmp.algorithm=hill_climb \
    --pass-config tir.disable_storage_rewrite=1 \
    --pass-config tir.disable_vectorize=1 \
    ocr_en/inference.pdmodel \
    --output-format=mlf \
    --model-format=paddle \
    --module-name=rec \
    --input-shapes x:[1,3,32,320] \
    --output=rec.tar
```

arm

VOICES on arm

TECH TALKS

Deployment on Arm Virtual Hardware

Arm Virtual Hardware

Enabling Software-Hardware Co-Design to Accelerate IoT and ML Development.

- Arm Virtual Hardware (AVH) scales and accelerates IoT software development by virtualising popular IoT development kits, Arm-based processors, and systems in the cloud.
- It is an evolution of Arm's modelling technology that removes the wait for hardware and the complexity of building and configuring board farms for testing.
- It enables modern agile software development practices, such as DevOps and MLOps workflows.

Multiple modeling technologies fitting a variety of use cases

Arm Virtual Hardware Corstone and CPUs

Cloud-based models of Corstone and Cortex-M processors for software development. Available via AWS.

We use this type in today's talk

Arm Virtual Hardware 3rd Party Hardware

Cloud-based models of popular IoT development kits, including peripherals, sensors and board components that are already in production. Available via hypervisor technology.

arm

Arm Virtual Hardware Corstone and CPUs

View Product Overview document to unlock more use cases

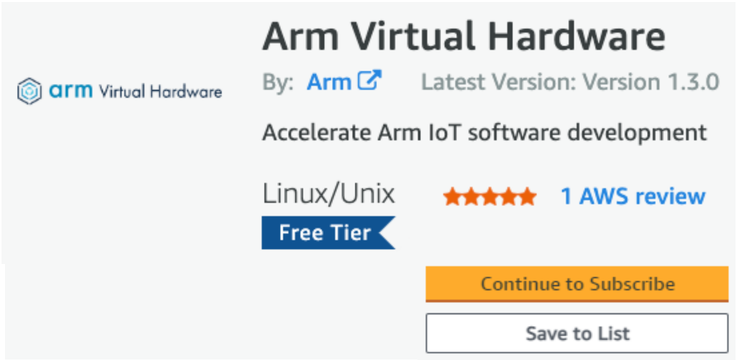
- Based on Arm Fast Model technology developed alongside Arm's processor IP
- Precisely simulates instruction and exception behaviors
- Offers test interfaces for the Open-CMSIS-CDI standard
- Runs on local hosted development systems as well as cloud-based CI/CD configurations
- Provides a scalable and extensible platform through SystemC

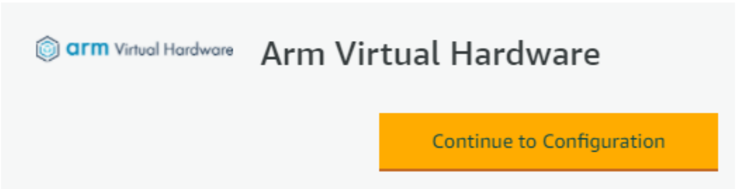
- Products included:
  - Corstone platforms: Corstone-300, Corstone-310 and Corstone-1000
  - Cortex-M processors: Cortex-M0, Cortex-M0+, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-M23, Cortex-M33
  - Note: You can check all supported virtual hardware under */opt* directory

Prerequisite

Reference steps to launch an Arm Virtual Hardware Amazon Machine Image (AMI) instance

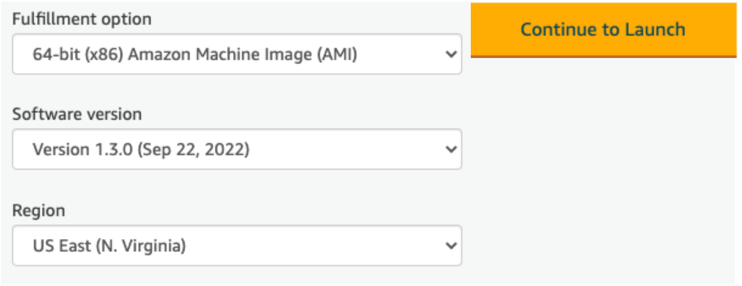
Step1. Visit AWS Marketplace or AWS China Marketplace. Subscribe to AVH and continue to configure.



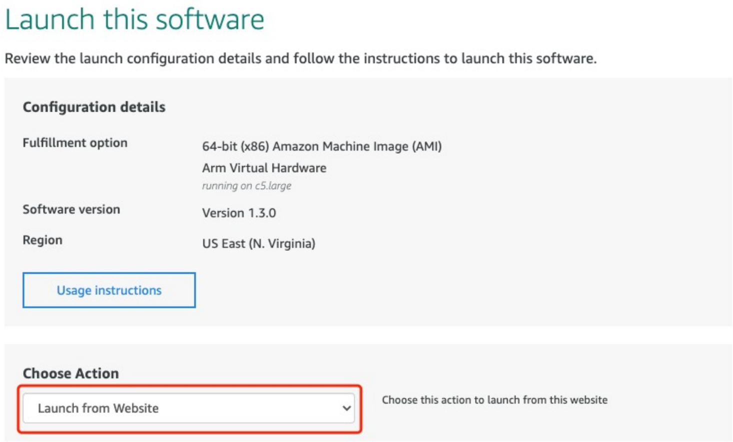


*Note: you will need an AWS/AWS China account as the prerequisite. Register an AWS account at https://aws.amazon.com.

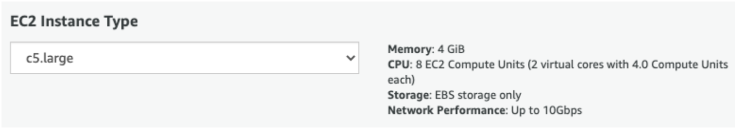
Step2. Choose region to deploy (server region) and continue to launch.

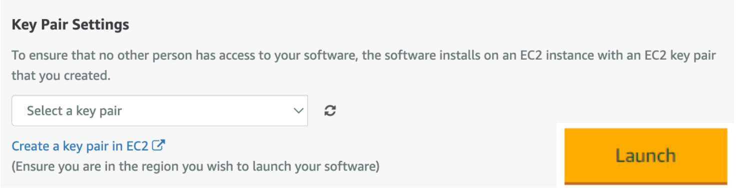


Step3. Choose "Launch from Website"(default).



Step4. Choose instance type, SSH key pair and launch. Then use `ssh-i key_pair ubuntu@public_ip` to log into the instance remotely.





Note:

1. Generally, “c5.large” instance type is recommended. For this example, you can try “t2.micro” which is free tier eligible.
2. Others default choice is ok.
3. If you don’t have a key pair, click “create a key pair in EC2” to create one.

Demo

Deploy PP-OCR Text Recognition Model on Arm Virtual Hardware

Demo Walkthrough

Build the application and deploy on Corstone-300 Platform (Cortex-M55 included)

- Fork and clone the sample code from GitHub. Navigate to the code path.

```
$ git clone https://github.com/ArmDeveloperEcosystem/Paddle-examples-for-AVH.git
(or from https://github.com/PaddlePaddle/PaddleOCR/tree/dygraph/deploy/avh)
$ cd Paddle-examples-for-AVH/OCR-example/
```

- The *run_demo.sh* script automates the entire process. It takes the following 6 steps to help you automatically build and execute the English text recognition application on Corstone-300 platform with Arm Virtual Hardware.
  - Step 1. Set up running environment.
  - Step 2. Download PaddlePaddle inference model.
  - Step 3. Use TVMC to compile the model and generate code for the Arm Cortex-M processor.
  - Step 4. Process resources for building the application image.
  - Step 5. Use the Makefile to build the target application.
  - Step 6. Run application binary on Corstone-300 platform integrated in AVH.

  Note: If you are not able to use AVH AMI hosted in AWS, you can use `--enable_FVP` to `1` to make the application run on local Corstone 300 FVP binary (`./run_demo.sh --enable_FVP 1`).

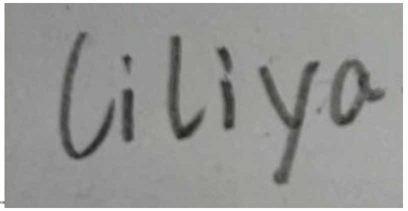
Results

- Test image: word_116.png
- Inference results on PC:
  - Predicts of path_to_word_116.png:('QBHOUSE', 0.9867456555366516).
- Inference results on AVH:

word_116.png

```
telnetterminal5: Listening for serial connection on port 5000
telnetterminal2: Listening for serial connection on port 5001
telnetterminal1: Listening for serial connection on port 5002
telnetterminal0: Listening for serial connection on port 5003

    Ethos-U rev 136b7d75 --- Feb 16 2022 15:47:15
    (C) COPYRIGHT 2019-2022 Arm Limited
    ALL RIGHTS RESERVED

Starting ocr rec inference
text: QBHOUSE, score: 0.986746
EXITTHESIM
Info: /OSCI/SystemC: Simulation stopped by user.
[warning ][main@0][01 ns] Simulation stopped by user

--- cpu_core statistics: ------------------------------------------------------
Simulated time                          : 76.450841s
User time                               : 99.066565s
System time                             : 0.009091s
Wall time                               : 100.220747s
Performance index                       : 0.76
cpu_core.cpu0                           :  19.29 MIPS (   1911271031 Inst)
--------------------------------------------------------------------------------
```

```
Starting ocr rec inference
text: Ciliya, score: 0.904513
EXITTHESIM
Info: /OSCI/SystemC: Simulation stopped by user.
[warning ][main@0][01 ns] Simulation stopped by user

--- cpu_core statistics: ------------------------------------------------------
Simulated time                          : 76.456830s
User time                               : 99.472250s
System time                             : 0.010264s
Wall time                               : 101.906402s
Performance index                       : 0.75
cpu_core.cpu0                           :  19.21 MIPS (   1911420761 Inst)
--------------------------------------------------------------------------------
```



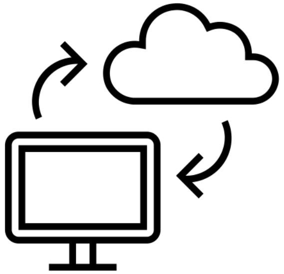
Inference results of rough handwriting(AVH)

arm

Next Steps

Join us to unlock more interesting use cases of PaddleOCR & Arm Virtual Hardware!

Register for free AWS EC2 Credits: bit.ly/AWS-Credits



Try the sample code in the tech talk and explore more use cases in Arm Virtual Hardware.

Get in touch to learn more
wangkai65@baidu.com
Liliya.wu@arm.com

Questions?

arm

arm

Tweet us: #ArmTechTalks

View tech talks on-demand:
www.youtube.com/arm

Sign up for upcoming tech talks:
www.arm.com/techtalks

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה