FIRENET - MODEL COMPRESSION

310552038 蔡瀚興
310554040 葉昭宏
309513017 黃梓熏

葉昭宏 310554040

What is the problem you want to explore?

在原本的論文中，提出FireNet model，而藉由另一篇論文

FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications 當中，可以知道FireNet能夠被應用在edge devices上，為了使model的size降低且不太降低accuracy的前提下

採用model quantization+ model pruning修改現有的model

What is your methodology?

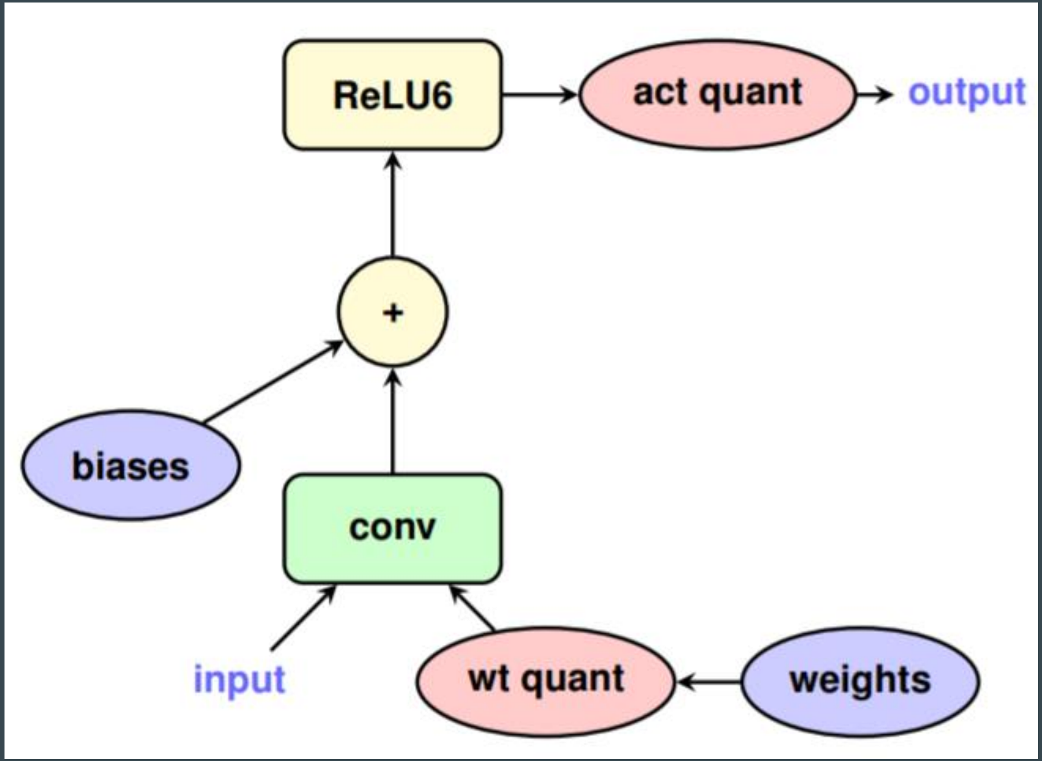
Quantization aware training

use post-training quantization and

quantization-aware training to optimize model

Introduction to Quantization aware training

In comparion with post-training, Quantization aware training can quantized specfic layer



Goal

1. Make FireNet model smaller than original

1. Make modified model's accuracy same or lower a little bit

What have you done?

1. test post-training quantization and quantization-aware result
2. test pruned + quantized model and find optimized combination
3. use keras tensorflow_model_optimization to control quantized layer

Result of my work

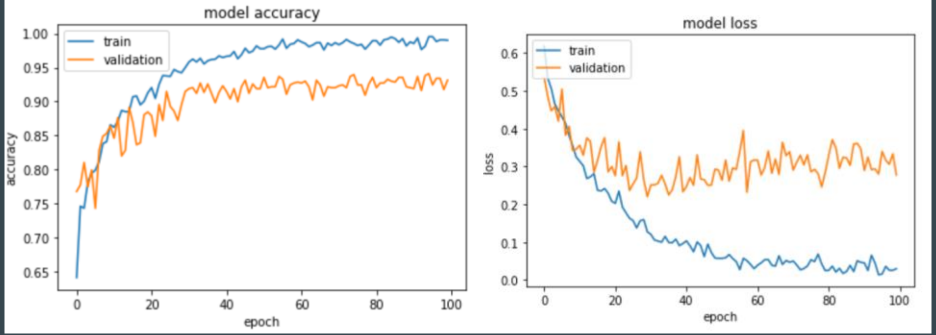
Baseline model detail

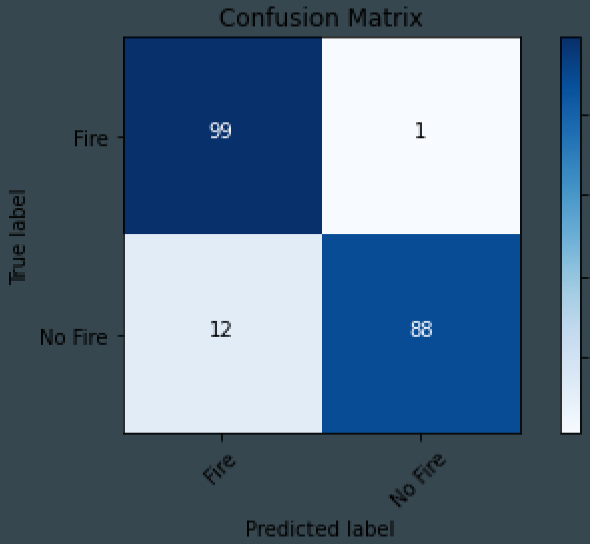
Precision: 0.89189

Recall: 0.99

Accuracy: 0.935

Size of gzipped baseline Keras model: 7061894.00 bytes





Post-training part

Average FPS : 45.313680281 67889(so low)

Accuracy : 0.935

Baseline model
Size: 7061894

Size of gzipped post-training TFlite model: 529885.00 bytes

Even if this training model size is the smallest model among many example, for the reason of speed, we didn't take post-training in consideration

Default quantization-aware training part

Average FPS : 243.55

Accuracy : 0.5

Baseline model
Size: 7061894

Size of gzipped post-training TFlite model: 761781.00 bytes

Dense layer + 8 bit quantized

Average FPS : 296.10078034062116(moderate speed)

Accuracy : 0.935 (we chose for result)

Size of gzipped 8bitDense TFlite model: 815270.00 bytes

Baseline model
Size: 7061894

Conv2D layer + 8 bit quantized

Average FPS : 436.98

Accuracy : 0.5

Size of gzipped 8bitConv2D TFlite model: 2352752.00 bytes

Baseline model
Size: 7061894

In this model, we found that the speed is the fastest, but the accuracy is a disaster

Dense layer + 8 bit quantized + pruned(best pruning)

Average FPS : 291.44

Accuracy : 0.91

Size of gzipped 8bitConv2D TFlite model: 982336.00 bytes

Baseline model
Size: 7061894

Conclusion in quantization

Best combination in quantization:

Dense layer only + 8bit quantization (moderate speed + moderate accuracy)

(we will show result in rasperry pi part)

There is a trade-off between pruning and quantization

310552038 蔡瀚興

What is the problem you want to explore?

原本論文提出的Firenet model，效率及準確率都還可以改進。

在Google Colab中跑論文給的model，結果如下：

FPS ≒ 12 (per sec)

Accuracy ≒ 0.935

Model weight size ≒ 7061894.00 (bytes)

因此我們希望能夠找出方法，增加FPS及縮小Model weight size，甚至提高Accuracy。

What is your methodology?

Magnitude-based weight pruning

- After each training epochs, the link with the smallest weight is removed.
- Thus the saliency of a link is just the absolute size of its weight.
- Though this method is simple, it rarely yields worse results than the more complicate algorithms.

What have you done?

1. Find dataset and build the Colab training environment
2. Write the python script for testing result on Colab and Rasberry Pi.
3. Find & try different model pruning strategy to optimize the result.

Result of my work

BASELINE MODEL

Hyperparameter

Batch = 32
Epochs = 100
Loss = sparse_categorical_crossentropy
optimizer = adam

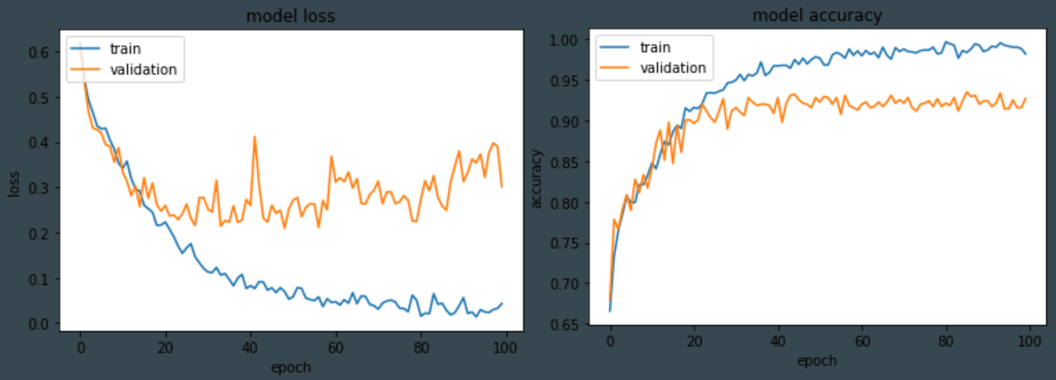
Dataset

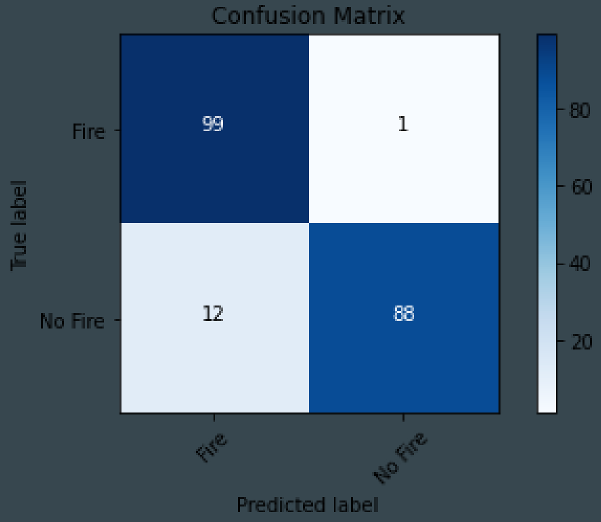
Fire : 1124
NoFire : 1301

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 62, 62, 16)
average_pooling2d (AveragePooling2D)	(None, 31, 31, 16)
dropout (Dropout)	(None, 31, 31, 16)
conv2d_1 (Conv2D)	(None, 29, 29, 32)
average_pooling2d_1 (AveragePooling2D)	(None, 14, 14, 32)
dropout_1 (Dropout)	(None, 14, 14, 32)
conv2d_2 (Conv2D)	(None, 12, 12, 64)
average_pooling2d_2 (AveragePooling2D)	(None, 6, 6, 64)
dropout_2 (Dropout)	(None, 6, 6, 64)
flatten (Flatten)	(None, 2304)
dense (Dense)	(None, 256)
dropout_3 (Dropout)	(None, 256)
dense_1 (Dense)	(None, 128)
dense_2 (Dense)	(None, 2)

BASELINE MODEL

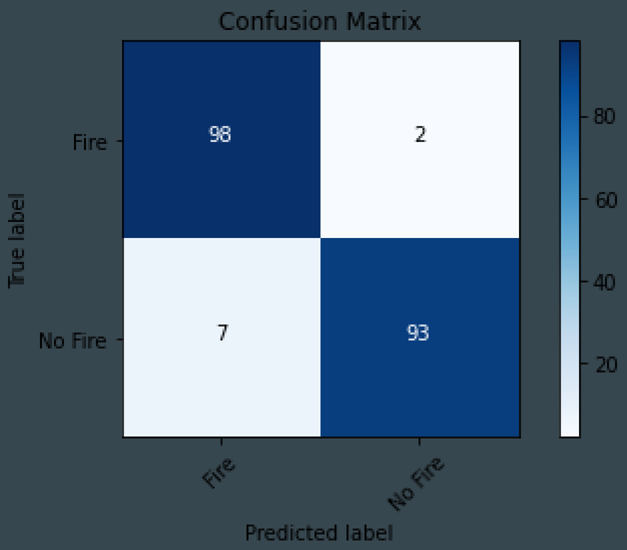
Precision: 0.89189
Recall: 0.99
Accuracy: 0.935





Size of gzipped baseline Keras model: 7061894.00 bytes

PRUNING ONLY DENSE LAYER



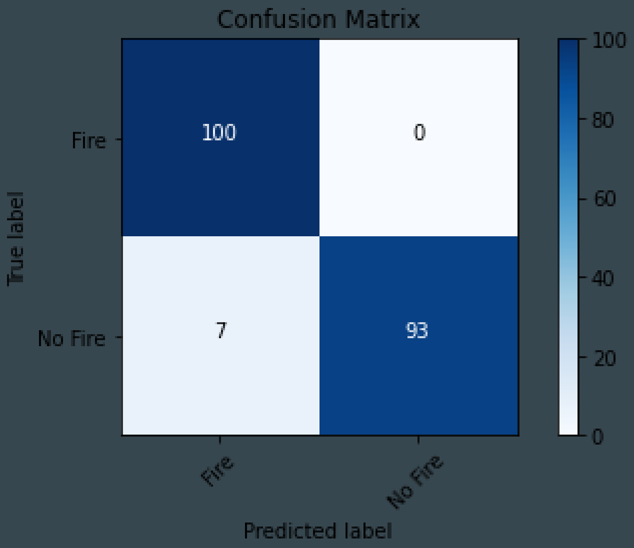
Precision: 0.93333
Recall: 0.98
Accuracy: 0.955

Baseline model
Accuracy: 0.935

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 62, 62, 16)
average_pooling2d (AveragePooling2D)	(None, 31, 31, 16)
dropout (Dropout)	(None, 31, 31, 16)
conv2d_1 (Conv2D)	(None, 29, 29, 32)
average_pooling2d_1 (AveragePooling2D)	(None, 14, 14, 32)
dropout_1 (Dropout)	(None, 14, 14, 32)
conv2d_2 (Conv2D)	(None, 12, 12, 64)
average_pooling2d_2 (AveragePooling2D)	(None, 6, 6, 64)
dropout_2 (Dropout)	(None, 6, 6, 64)
flatten (Flatten)	(None, 2304)
prune_low_magnitude_dense (PruneLowMagnitude)	(None, 256)
dropout_3 (Dropout)	(None, 256)
prune_low_magnitude_dense_1 (PruneLowMagnitude)	(None, 128)
prune_low_magnitude_dense_2 (PruneLowMagnitude)	(None, 2)

Size of gzipped baseline Keras model : 7061894.00 bytes
Size of gzipped pruned Keras model : 1536520.00 bytes

PRUNING ONLY CONV LAYER



Precision: 0.93458
Recall: 1.0
Accuracy: 0.965

Baseline model
Accuracy: 0.935

Layer (type)	Output Shape
prune_low_magnitude_conv2d (PruneLowMagnitude)	(None, 62, 62, 16)
average_pooling2d (AveragePooling2D)	(None, 31, 31, 16)
dropout (Dropout)	(None, 31, 31, 16)
prune_low_magnitude_conv2d_1 (PruneLowMagnitude)	(None, 29, 29, 32)
average_pooling2d_1 (AveragePooling2D)	(None, 14, 14, 32)
dropout_1 (Dropout)	(None, 14, 14, 32)
prune_low_magnitude_conv2d_2 (PruneLowMagnitude)	(None, 12, 12, 64)
average_pooling2d_2 (AveragePooling2D)	(None, 6, 6, 64)
dropout_2 (Dropout)	(None, 6, 6, 64)
flatten (Flatten)	(None, 2304)
dense (Dense)	(None, 256)
dropout_3 (Dropout)	(None, 256)
dense_1 (Dense)	(None, 128)
dense_2 (Dense)	(None, 2)

Size of gzipped baseline Keras model : 7061894.00 bytes
Size of gzipped pruned Keras model : 2377961.00 bytes

PRUNING ALL LAYER

Three pruning strategy：

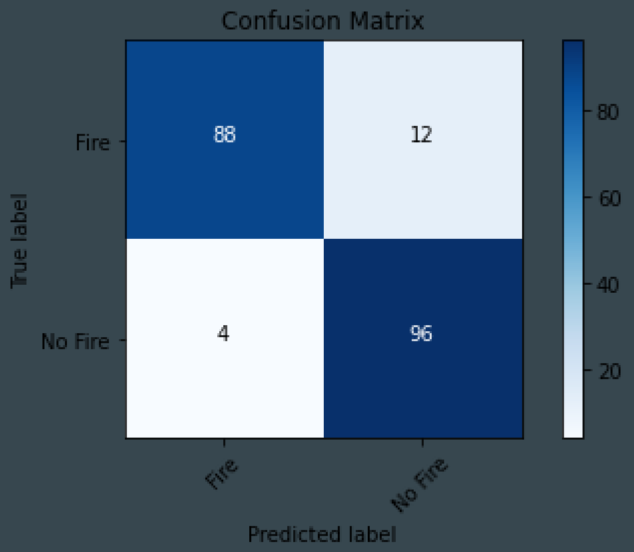
- PolynomialDecay from 100% ~ 70%
- PolynomialDecay from 100% ~ 20%
- ConstantSparsity with 50%

Layer (type)	Output Shape
prune_low_magnitude_conv2d (PruneLowMagnitude)	(None, 62, 62, 16)
prune_low_magnitude_average_pooling2d (PruneLowMagnitude)	(None, 31, 31, 16)
prune_low_magnitude_dropout (PruneLowMagnitude)	(None, 31, 31, 16)
prune_low_magnitude_conv2d_1 (PruneLowMagnitude)	(None, 29, 29, 32)
prune_low_magnitude_average_pooling2d_1 (PruneLowMagnitude)	(None, 14, 14, 32)
prune_low_magnitude_dropout_1 (PruneLowMagnitude)	(None, 14, 14, 32)
prune_low_magnitude_conv2d_2 (PruneLowMagnitude)	(None, 12, 12, 64)
prune_low_magnitude_average_pooling2d_2 (PruneLowMagnitude)	(None, 6, 6, 64)
prune_low_magnitude_dropout_2 (PruneLowMagnitude)	(None, 6, 6, 64)
prune_low_magnitude_flatten (PruneLowMagnitude)	(None, 2304)
prune_low_magnitude_dense (PruneLowMagnitude)	(None, 256)
prune_low_magnitude_dropout_3 (PruneLowMagnitude)	(None, 256)
prune_low_magnitude_dense_1 (PruneLowMagnitude)	(None, 128)
prune_low_magnitude_dense_2 (PruneLowMagnitude)	(None, 2)

PolynomialDecay from 100% ~ 70%

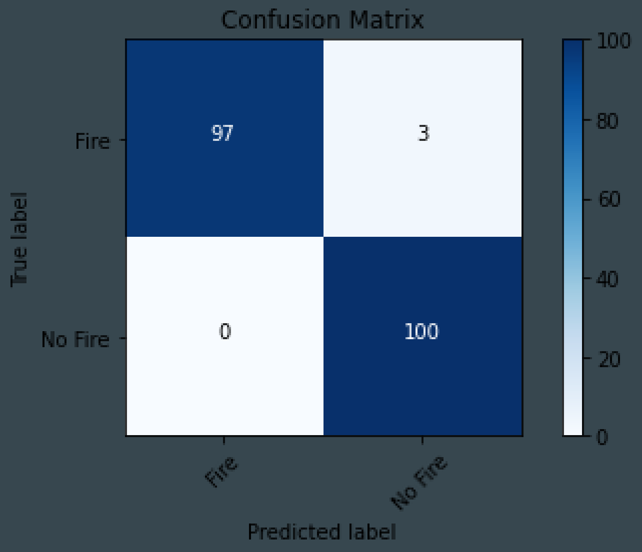
Precision: 0.95652
Recall: 0.88
Accuracy: 0.92

Baseline model
Accuracy: 0.935



- Size of gzipped baseline Keras model : 7061894.00 bytes
- Size of gzipped pruned Keras model : 774873.00 bytes

PolynomialDecay from 100% ~ 20%



Precision: 1.0
Recall: 0.97
Accuracy: 0.985

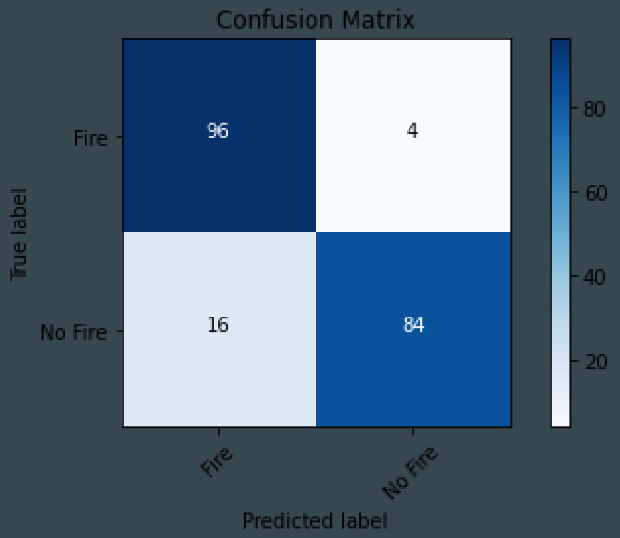
Baseline model
Accuracy: 0.935

- Size of gzipped baseline Keras model : 7061894.00 bytes
- Size of gzipped pruned Keras model : 772341.00 bytes

ConstantSparsity with 50%

Precision: 0.85714
Recall: 0.96
Accuracy: 0.9

Baseline model
Accuracy: 0.935



- Size of gzipped baseline Keras model : 7061894.00 bytes
- Size of gzipped pruned Keras model : 1513879.00 bytes

309513017 黃梓熏

What is the problem you want to explore?

嘗試將Model Pruning與Quantization結合，並同時套用到Firenet model上，來觀察是否能讓model在保持與Baseline model(只單用Model Pruning的model)差不多的準度的情況下，model的size有大幅降低的效果．

What is your methodology?

Pruning preserving quantization aware training (PQAT)

→ 是tensorFlow底下提供的一種優化方法，能讓做過pruning的model在為持原本pruning的效果下再去做quantization。

What have you done?

1. Combine pruning and quantization into Firenet model by using PQAT
2. Setup the execution environment for Raspberry Pi 4b
3. Test the performance of tensorFlow lite model on Raspberry Pi 4b

Result of my work

PQAT

Pruning method：PolynomialDecay from 100% ~ 20%

Quantization method：8 bits prune preserve quantization

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 16)        448

 average_pooling2d (AverageP  (None, 31, 31, 16)       0
 ooling2D)

 dropout (Dropout)           (None, 31, 31, 16)        0

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        4640

 average_pooling2d_1 (Averag  (None, 14, 14, 32)       0
 ePooling2D)

 dropout_1 (Dropout)         (None, 14, 14, 32)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        18496

 average_pooling2d_2 (Averag  (None, 6, 6, 64)         0
 ePooling2D)

 dropout_2 (Dropout)         (None, 6, 6, 64)          0

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 256)               590080

 dropout_3 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 2)                 258

=================================================================
Total params: 646,818
Trainable params: 646,818
Non-trainable params: 0
```

Model Size

- Size of gzipped origin Keras model : 7061894.00 bytes
- Size of gzipped pruned Keras model (baseline) : 772341.00 bytes
- Size of gzipped pruned TFlite model(baseline) : 754639.00 bytes
- Size of gzipped pruned TFlite model(with PQAT) : 225301.00 bytes

Accuracy

	Dataset 1	Dataset 2	Dataset 3
Baseline model (Keras)	0.92	0.985	0.965
Baseline model (TFlite)	0.92	0.985	0.965
Model with PQAT (TFlite)	0.965	0.965	0.955

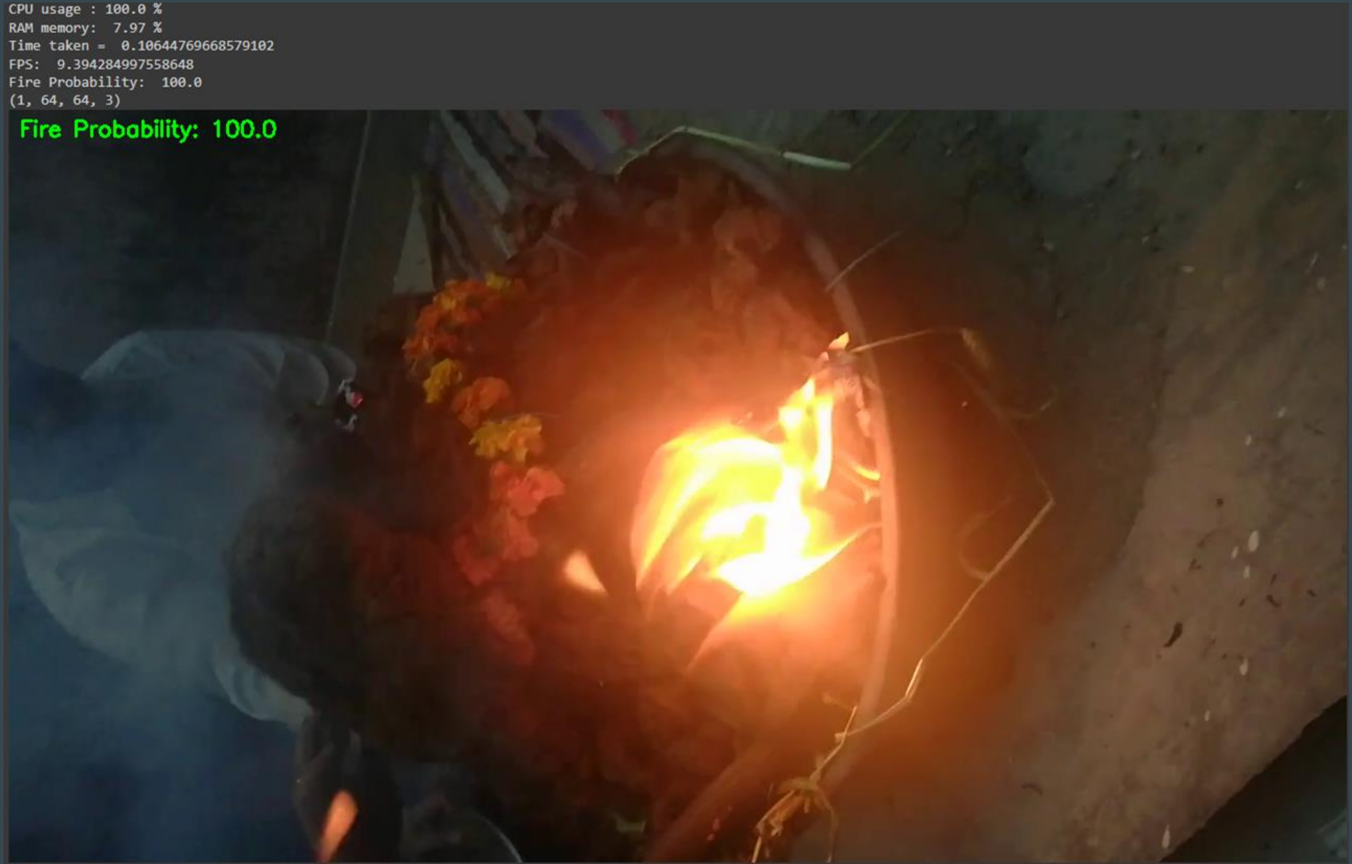
Result of our work

Baseline model

- We can’t run baseline model on Rasberry Pi is too lag
- The next page is the result of real time video run on Colab

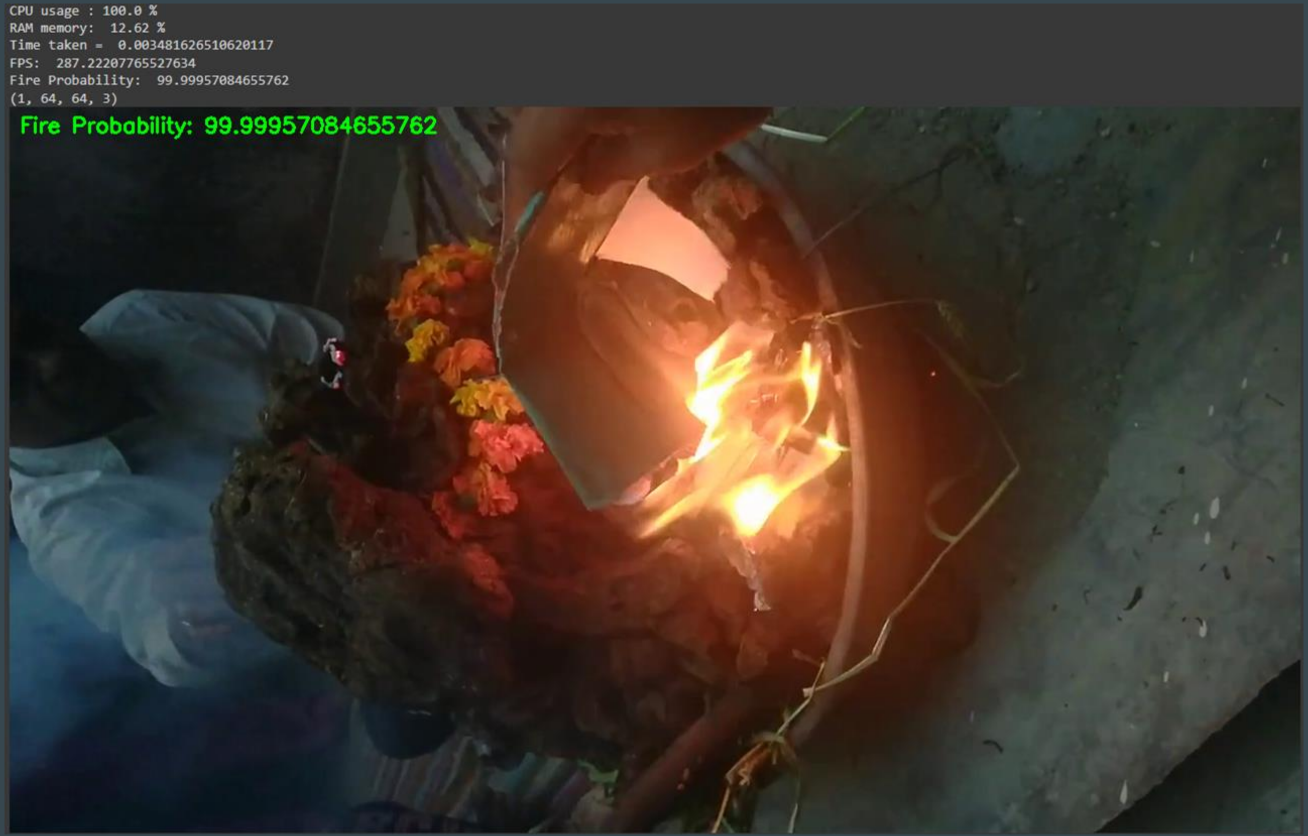
Run on Colab

Baseline model



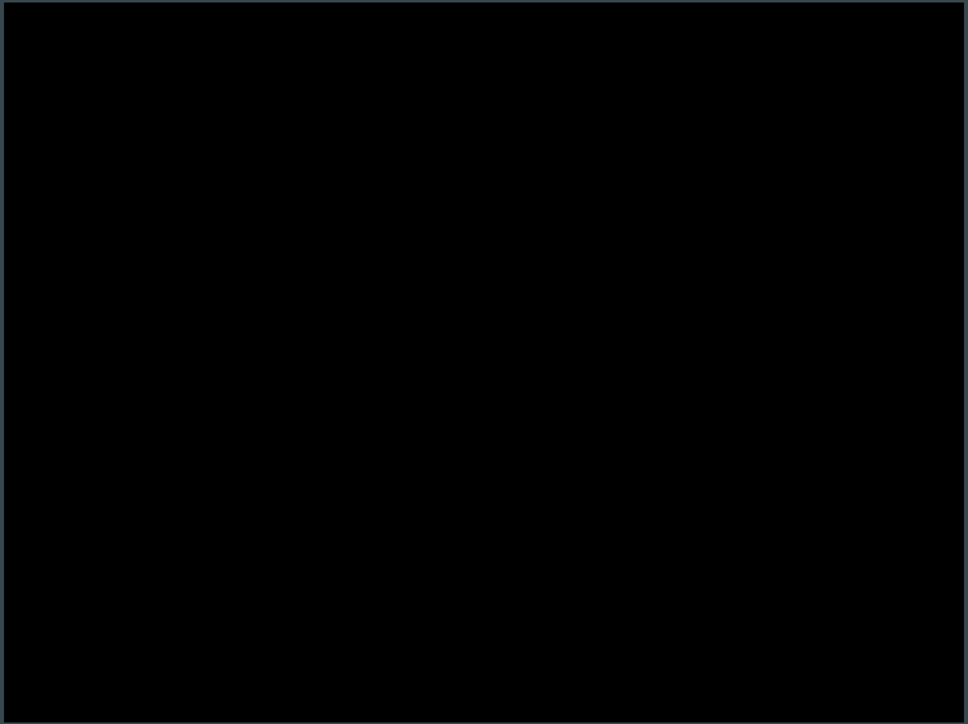
Run on Colab

Best pruning stratgy – PolynomialDecay from 100% ~ 20%



Run on Rasberry Pi

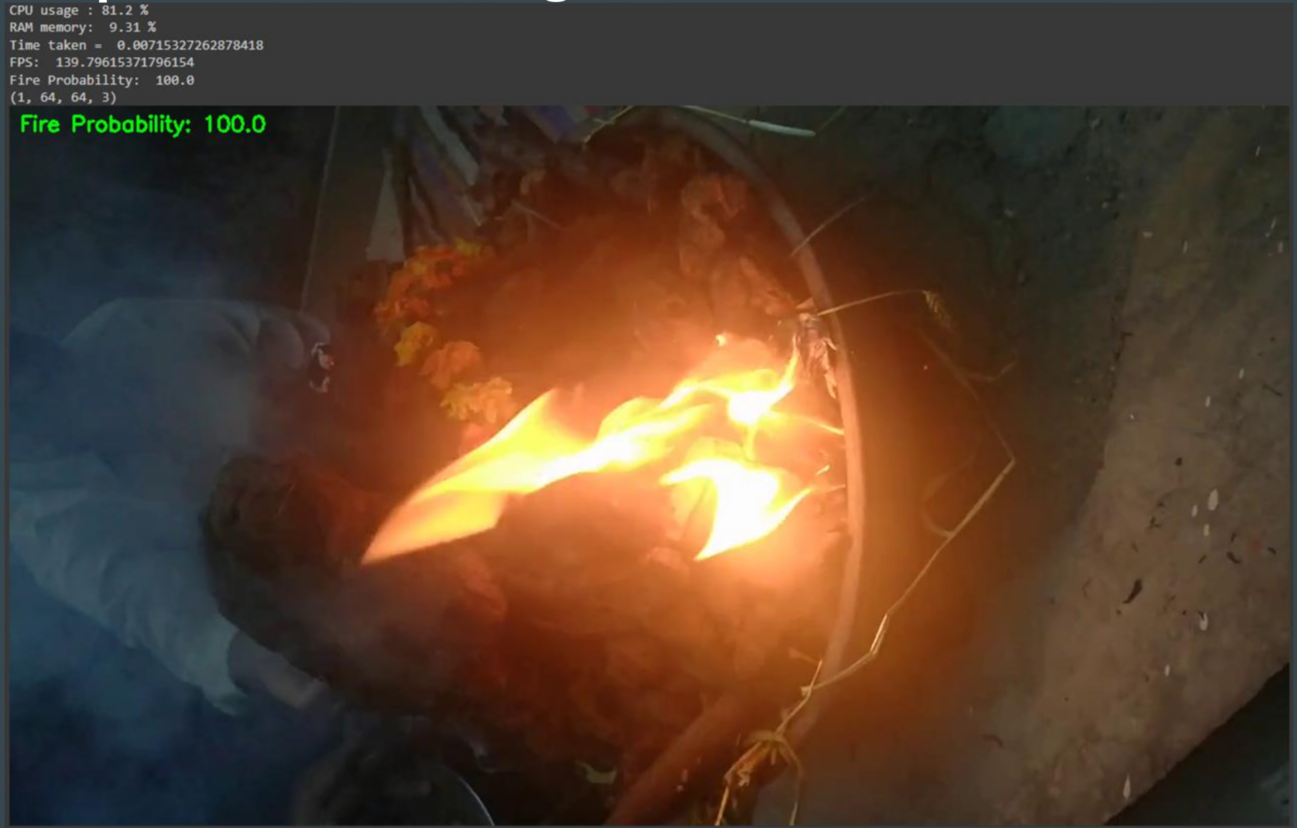
Best pruning stratgy – PolynomialDecay from 100% ~ 20%



http://drive.google.com/file/d/12SXF162-zVikSd3rwXMuh2kLwLngsqbS/view

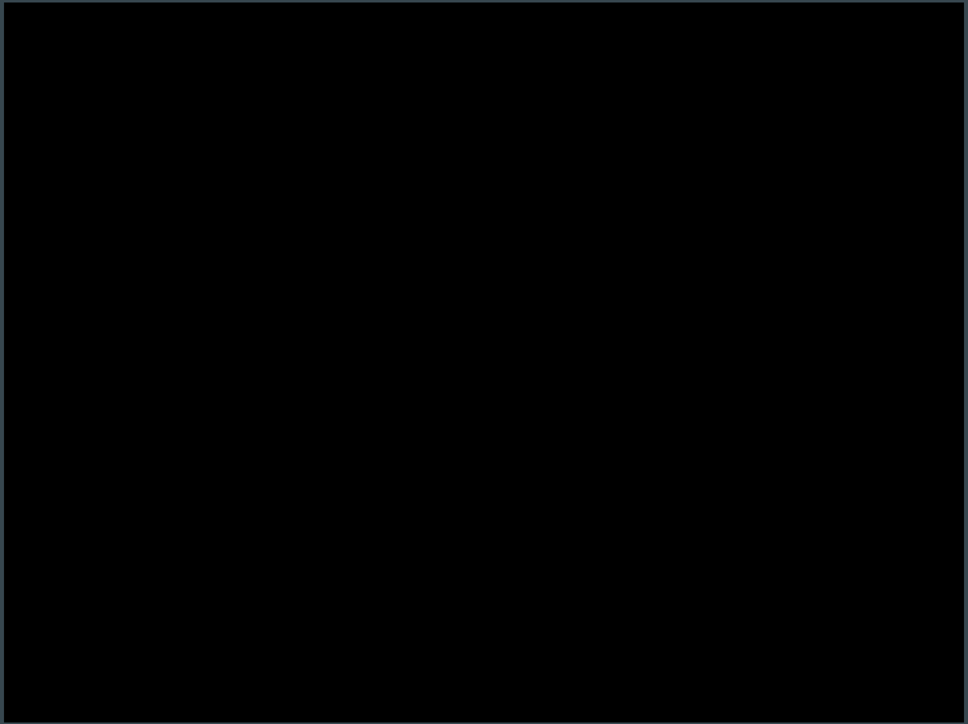
Run on Colab

Best quantization stratgy – int 8 bit on dense layer only



Run on Rasberry Pi

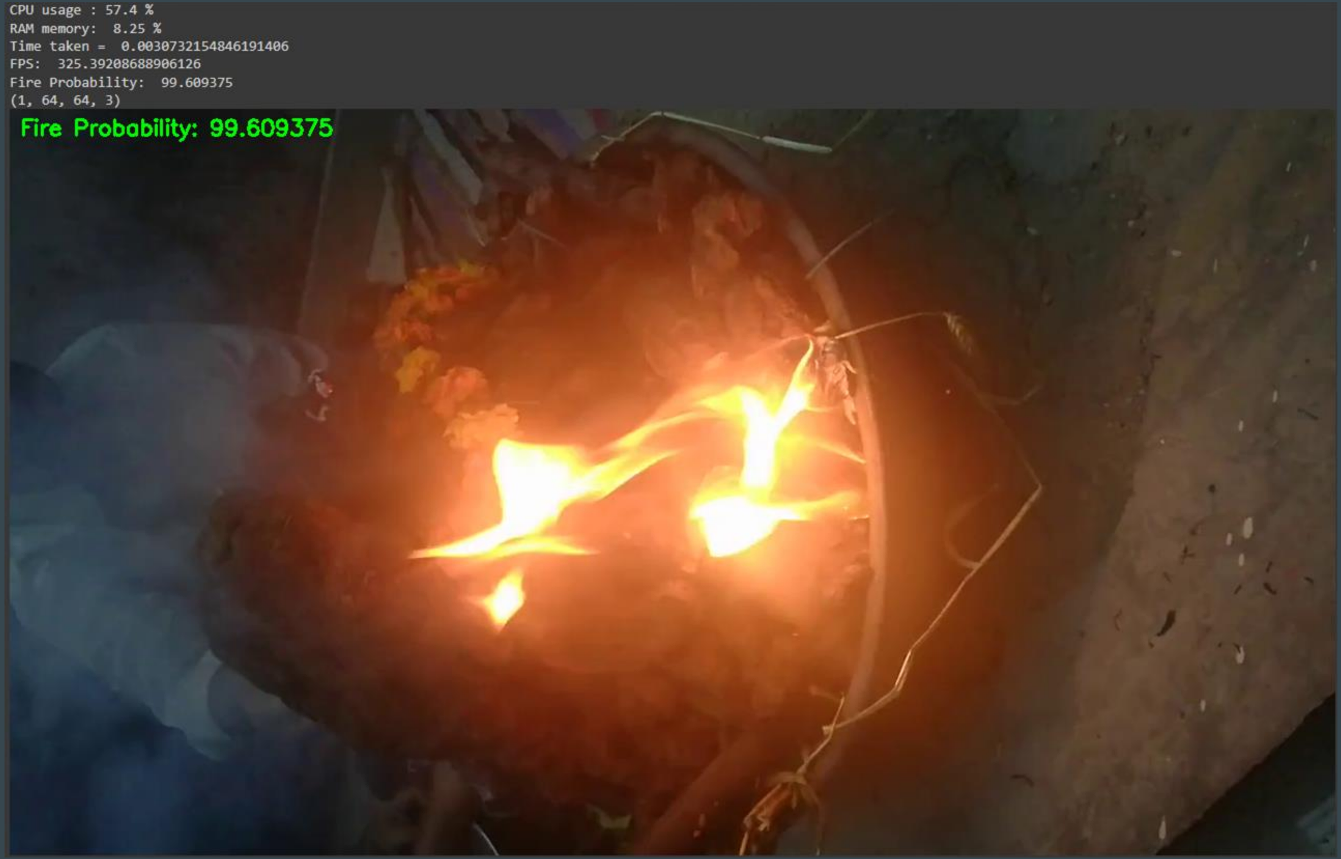
Best quantization stratgy – int 8 bit on dense layer only



http://drive.google.com/file/d/143fAWxbA8UcbP97n57CdQ2Rg7_BnZsJ7/view

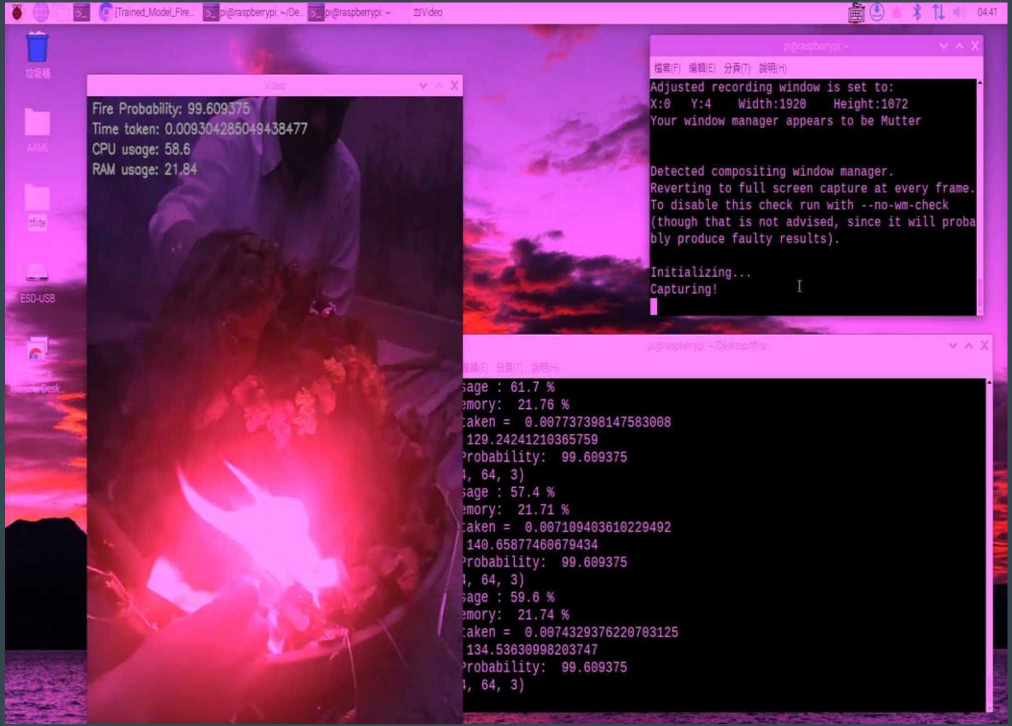
Run on Colab

Do best pruning strategy first then do best quantization strategy



Run on Rasberry Pi

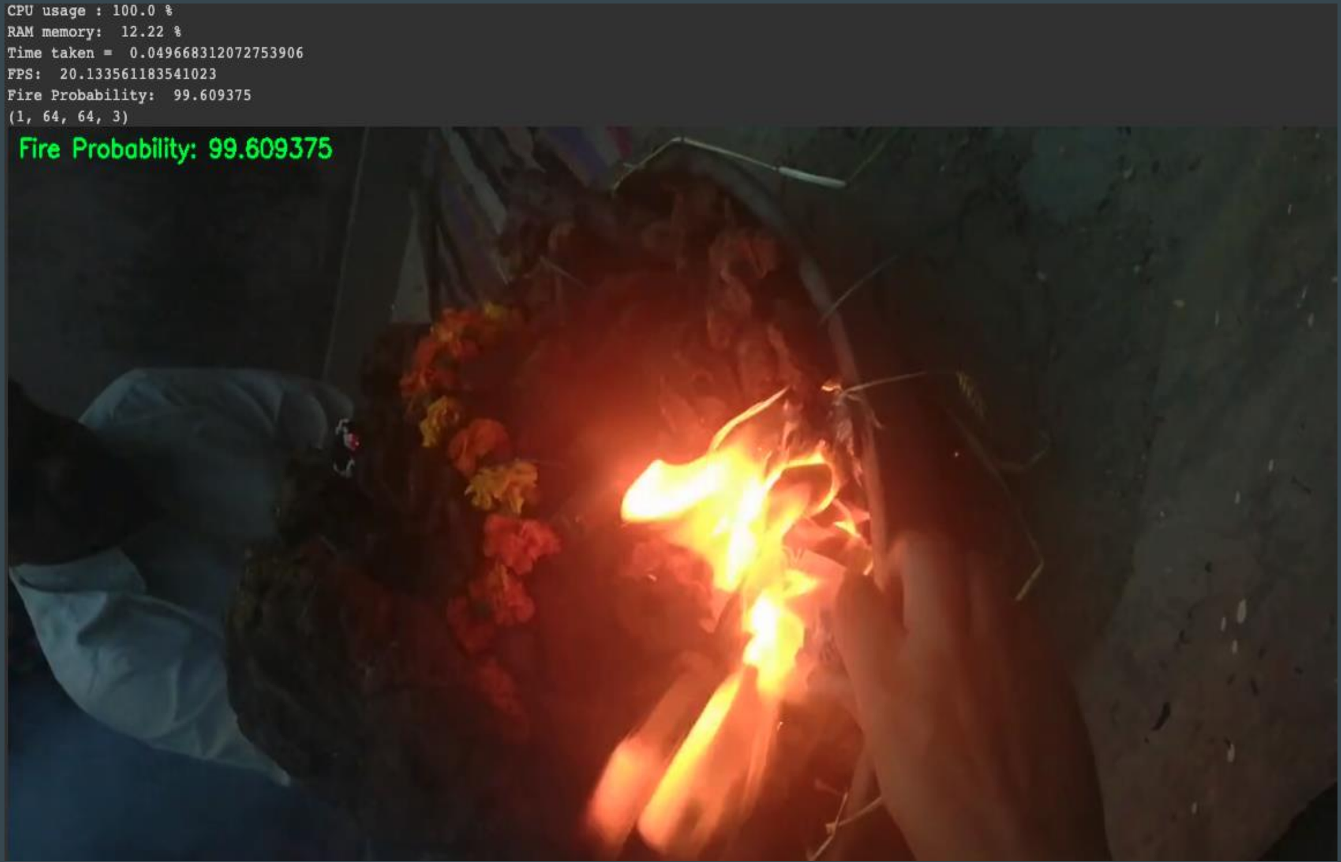
Do best pruning then best quantization



http://drive.google.com/file/d/1r18K1xDv7iAJEVCtRydJhLCgP2JJU5mM/view

Run on Colab

PQAT : PolynomialDecay from 100% ~ 20% + 8 bits prune preserve quantization



Run on Rasberry Pi

PQAT : PolynomialDecay from 100% ~ 20% + **8 bits prune preserve quantization**

http://drive.google.com/file/d/14qsNR3IXSFZVGxHdBL5GVJ96HlanJEFO/view