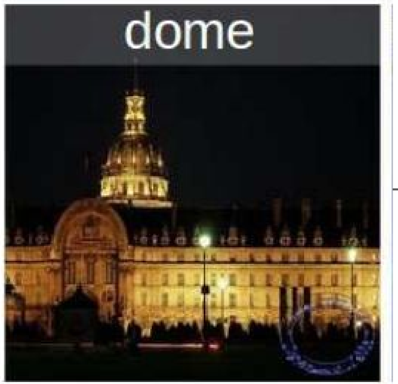
카카오 1차 면접

Presented by 서석준

Table of Contents

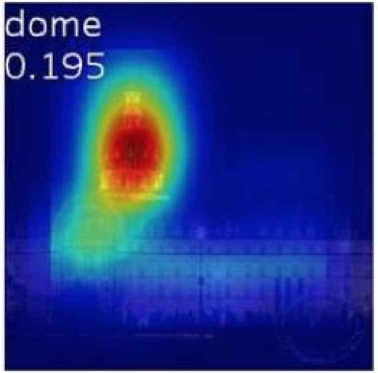
Weakly Supervised Object Detection (Localization)

- Compared to amount of image data, number of well annotated objects is very small.
- Let’s only utilize image-level annotation while predicting the position of the objects.



MODEL

Only trained on image-level annotation



Hide-and-Seek (HAS)

- Hide-and-Seek: Forcing a Network to be Meticulous for Weakly-supervised Object and Action Localization.
    - Krishna Kumar Singh and Yong Jae Lee.
        - University of California, Davis.
        - ICCV 2017.
    - Simple but powerful.
        - According to the paper.

- Main Idea
    - CAM (Class Activation Mapping) based localization showed good performance.
    - However, previous method is tend to concentrate on the most discriminative part.
        - Which means that,
        - It can lead to too tight bounding box.
    - If we hide some patches of input, a model might learn more generalized parts of object.
    - It only manipulates the input image!
        - So can be applied to any model.

Learning Deep Features for Discriminative Localization

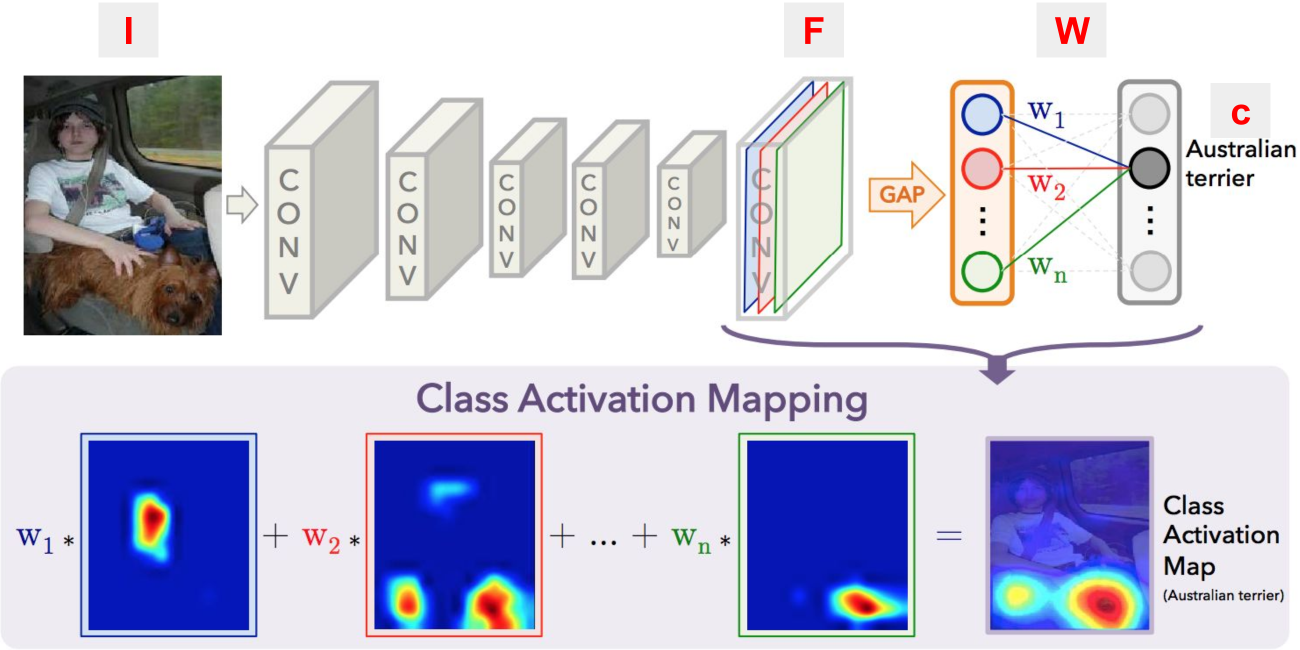
- Learning Deep Features for Discriminative Localization
  - Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba.
    - MIT.
    - CVPR 2016.
  - Previous study of Hide-and-Seek.
  - Introducing CAM.

- Features
  - CAM can be used to object localization.
  - How to make bounding box from CAM?
    - Largest connected component of binary segmentation map.
    - For image X and threshold θ,

$$map_{i,j} = \begin{cases} fg & if \;\; x_{i,j} \geq \theta \max(x) \\ bg & if \;\; x_{i,j} < \theta \max(x) \end{cases}$$

  - Can adapt any CNN based image classification model.

CAM (Class Activation Mapping)



$F=\{F_1,\ldots,F_M\}$: M *feature maps*

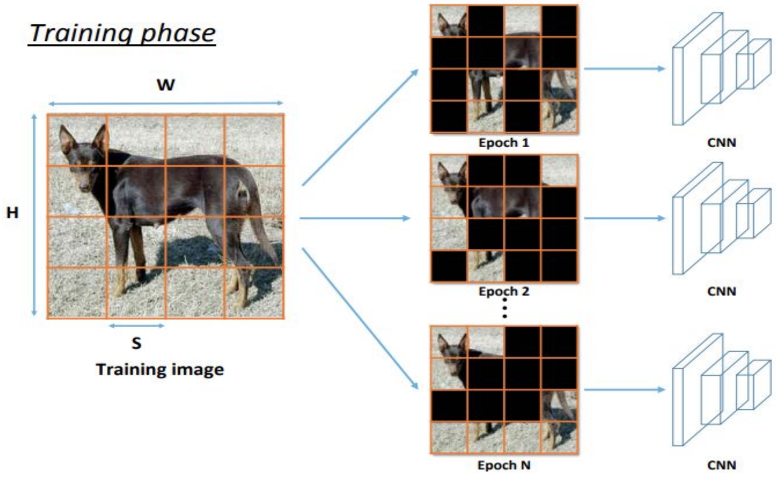
W: *last convolution layer*

c: *class*, I: *image*

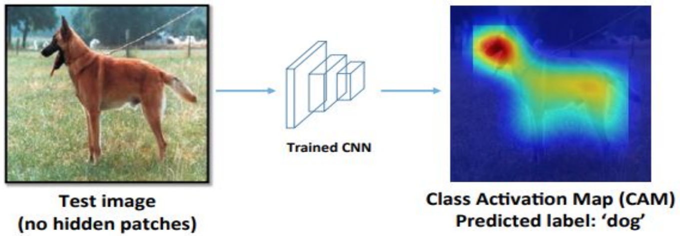
$$CAM(c,I)=\sum_{i=1}^{M} W(c,i)\cdot F_i(I)$$

Hide-and-Seek

Training phase



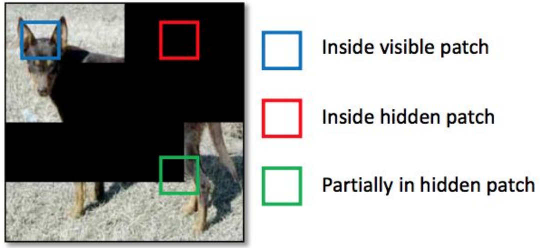
Testing phase



$$W=\{w_1,\ldots,w_{k\times k}\}$$

$$X=\{x_1,\ldots,x_{k\times k}\}$$

$$\mu=\frac{1}{N_{pixels}}\sum_j x_j$$



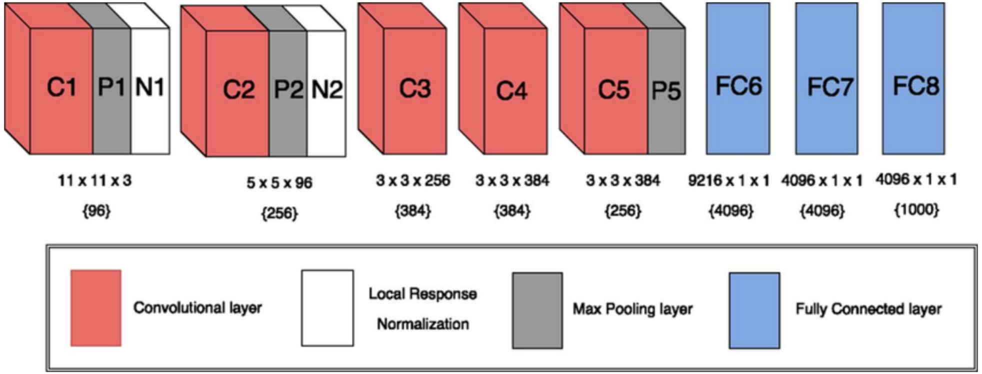
$$Conv(W,X)=\begin{cases}\sum_{i=1}^{k\times k} w_i^T x_i \\ \sum_{i=1}^{k\times k} w_i^T \mu \\ \sum_{m\in visible} w_m^T x_m + \sum_{n\in hidden} w_n^T \mu \approx \sum_{i=1}^{k\times k} w_i^T \mu\end{cases}$$

- Randomly hiding patches of training images.
  - Should be different for each epoch.
  - Also should be different within a batch.
- When hiding, It is important to fill the values with mean value.
  - In terms of normalization!

Implementation - from scratch

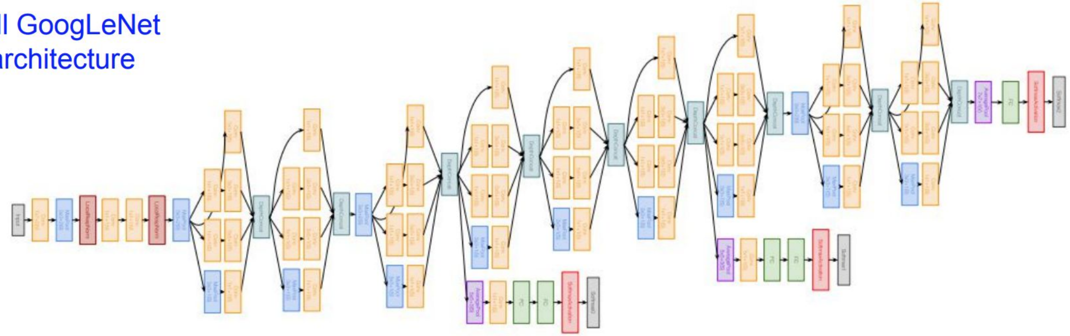
- Let’s implement from scratch!
    - There is no available source code :D
    - For previous study, only matlab implementation is available.
    - There exists Tensorflow version but is neither official nor sufficient.
- To make CAM, it is essential to build Alexnet and Googlenet first.
    - Both paper use Alexnet and Googlenet as the classification model.

Alexnet & Googlenet



- Alexnet
- ReLU, local normalization, weight decay.
- 60M parameters.

Full GoogLeNet architecture



- Googlenet
- Introduce inception module.
- 60M parameters.
- 12x less parameters than Alexnet.

22 total layers with weights (including each parallel layer in an Inception module)

Alexnet & Googlenet + GAP

```python
### additional conv layer
with tf.variable_scope('conv_5'):
  x = net.conv_relu_bn(x,
                       filter_shape=[3,3],
                       num_filters=1024,
                       stride=1,
                       padding='SAME',
                       is_training=self.is_training,
                       regularizer=None)
  self.F = x
  # x: [batch, 14, 14, 1024]
  # summaries.append( tf.summary.histogram('conv_5', x) )

### GAP
# [batch, 14, 14, 512] => [batch, 1024]
with tf.variable_scope('gap'):
  # x = tf.reduce_sum(x, axis=[1, 2])
  x = tf.reduce_mean(x, axis=[1, 2])
  # x: [batch, 1024]
  # summaries.append( tf.summary.histogram('gap', x) )

### softmax without bias
with tf.variable_scope('softmax'):
  num_feature = x.get_shape().as_list()[-1]
  weights = tf.get_variable("weights",
              shape=[num_feature, self.num_classes],
              initializer=tf.contrib.layers.xavier_initializer(),
              regularizer=None)

  self.logits = tf.matmul(x, weights)
  # logits: [batch, num_classes]
```

- Implemented at *src/alexnet_gap.py* and *src/googlenet_gap.py*
- According to Hide-and-Seek paper.
  - From base models, detach all layers after specific pooling layers.
  - Attach additional convolution layer followed by GAP and softmax layer.
  - Attach batch normalization layer after all convolution layers.
- Using pre-trained model?
  - Since layer architecture is slight different, it is hard to used pre-trained model.
  - Also, both papers commented that pre-trained did not improve performance.
  - Therefore, all models are trained from scratch.

Alexnet & Googlenet GAP

- Image size issue.
  - Both model is designed to get 227x227 and 224x224 images.
  - However, our image size is 64x64.
  - Upscale image to 227 and 224 respectively.

Will be discussed more.

- Papers are not kind…
  - If there exist conflicts between two papers, follow the setting of HAS paper.
    - # neurons in of alexnet gap layer => 512.
    - θ (segmentation threshold) => 20% / 30%.
  - Not describing the details of experiment setting.
    - Optimizer?
      - SGD + Momentum (0.9) since Alexnet and Googlenet use that setting.
    - Weight decay of Alexnet?
      - Not mentioned. So applied same value (0.0005) used for original Alexnet.
    - LR is gradually decreased from 0.01 to 0.0001?
      - No details. So use linear interpolation.

Alexnet & Googlenet GAP + HAS

- Adding HAS to GAP models.
  - Hide random patches at train time while do 10 multi-crop test at test time.
    - Rate of crop is not mentioned. So I naively crop 75% of image since large crop might lose too much information as our image is quite small.
  - Averaging all 10 crops to make the CAM and probability of classes.
- Why using multi-crop?
  - Also not mentioned.
  - Will be discussed more!

Alexnet & Googlenet GAP + HAS

- 3 Evaluation metrics.
  - Top-1 class accuracy.
    - Classification accuracy of top-1 prediction.
  - GT-known localization accuracy.
    - Fraction of more than 50% IoU (Jaccard coefficient) between predicted bounding box of ground-truth class and ground-truth box of ground-truth class.
    - Most focused on!
  - Top-1 localization accuracy.
    - Top-1 && GT-known localization.

Other Issues

- Grayscale Images.
    - 1.8% of images are grayscale. Change them to RGB by duplicating values.
- Image normalization.
    - Not mentioned. However, without it, performance is too bad.
    - Mean normalization without rescaling.
- Approximation for partially hidden patches.
    - It works like dilating hiding patches.
    - 50% is already hidden but even more dilation?
    - Implemented without approximation.

$$\sum_{m \in visible} w_m^T x_m + \sum_{n \in hidden} w_n^T \mu \approx \sum_{i=1}^{k \times k} w_i^T \mu$$

Other Issues

- Number of patches reported on the paper is weird.
    - They used 16, 32, 44, 56 number of patches.
    - However, is it possible?

> Concretely, given a training image I of size $W \times H \times 3$, we first divide it into a grid with a fixed patch size of $S \times S \times 3$. This results in a total of $(W \times H)/(S \times S)$ patches. We

$$S = \sqrt{\frac{W \times H}{N}} = \frac{W}{\sqrt{N}}$$

        - S cannot be an integer.
        - Therefore, I used 16, 36, 64, mixed number of patches.
- Train epoch.
    - Even though the paper trained for 55 and 40 epochs, I found that the loss keeps decrease so set epochs as 200 and 100.
    - Can be caused by different optimizer.
- Training is quite slow.
    - As I don't have enough machines, most of the experiments were done with Alexnet.

Experiment Overview

- I tried to reproduce as many experiments as possible.
    a. Basic experiment.
    b. Dropout.
    c. GMP.
- Moreover, I tried several more experiments which I thought important but not did in paper.
    a. Multi-Crop Test.
    b. Localization Threshold.
    c. Resizing Image and Network.
    d. Image Augmentation.
    e. Deeper Model.

Experiment 1

Basic Experiment

Experiment

- Dataset - Tiny ImageNet.
  - Small subset of ImageNet Challenge.
  - 200 classes, 500 train / 50 validation data for each class.
  - 64x64 Image.
- Basic experiment.
  - Comparing performance of Alex/GoogleNet-GAP with Alex/GoogleNet-HAS.
  - Do multi-crop test.
  - Localization threshold 0.2 / 0.3.

Result

Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	51.07	21.31	39.68
AlexNet-HAS-16	50.66	20.47	38.99
AlexNet-HAS-36	50.76	18.70	35.36
AlexNet-HAS-64	50.80	19.68	37.27
AlexNet-HAS-Mix	50.92	20.19	37.73
GoogleNet-GAP	53.28	24.79	43.44
GoogleNet-HAS-16	52.61	24.00	43.14

- Complex model showed better performance.
- However, it seems that HAS has no effect on performance.
  - Why?
  - Two hypotheses.
    - Because of multi-crop test.
    - Because of fixed and naive localization threshold.

Additional Experiment 1

Multi-crop Test

Experiment

- Is Multi-crop test essential?
  - Neither justification nor comparison in paper (also no details…).
    - Since I don’t know how much fractions are cropped, it is hard to reproduce same result.
  - So, let’s remove multi-crop test and compare the performances.
    - Same trained model.
    - Localization threshold 0.2 / 0.3.

Result

	With Multi-Crop			Without Multi-Crop		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	51.07	21.31	39.68	52.65	23.54	39.27
AlexNet-HAS-16	50.66	20.47	38.99	51.82	24.76	42.23
AlexNet-HAS-36	50.76	18.70	35.36	51.90	24.98	42.24
AlexNet-HAS-64	50.80	19.68	37.27	52.43	26.47	44.24
AlexNet-HAS-Mix	50.92	20.19	37.73	52.72	26.97	45.21
GoogleNet-GAP	53.28	24.79	43.44	55.47	30.22	47.63
GoogleNet-HAS-16	52.61	24.00	43.14	54.85	31.94	51.38

- It seems that naive multi-crop approach rather decreases performance.
  - Issue of Tiny Imagnet data.
- Therefore, for further experiments, I would exclude multi-crop test.

Additional Experiment 2

Localization Threshold

Experiment

- Localization threshold is fixed by authors to 0.2 and 0.3 for Alexnet and GoogleNet respectively.
  - They commented that they chose the values by hand.
    - Even though it is very important factor determining GT-known Loc and Top-1 Loc.
  - However, in our case, dataset is changed which means that the value might be harmful.
    - The authors already tested multiple thresholds for video set!
    - Let’s test for various thresholds = {0.2, 0.3, 0.4, 0.5}
    - Without multi-crop.

Result

	0.2			0.3			0.4			0.5			Best (GT)		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas	GT-known Loc	Top-1 Loc	Top-1 Clas	GT-known Loc	Top-1 Loc	Top-1 Clas	GT-known Loc	Top-1 Loc	Top-1 Clas	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	52.65	23.54	39.27	54.31	24.53	39.27	53.38	23.51	39.27	47.06	19.57	39.27	54.31	24.53	39.27
AlexNet-HAS-16	51.82	24.76	42.23	53.99	26.21	42.23	54.70	26.31	42.23	51.01	23.53	42.23	54.70	26.31	42.23
AlexNet-HAS-36	51.90	24.98	42.24	54.41	26.45	42.24	54.86	26.22	42.24	50.71	23.33	42.24	54.86	26.22	42.24
AlexNet-HAS-64	52.43	26.47	44.24	54.42	27.80	44.24	55.22	27.74	44.24	51.32	24.47	44.24	55.22	27.74	44.24
AlexNet-HAS-Mix	52.72	26.97	45.21	55.04	28.47	45.21	54.90	27.96	45.21	50.01	24.60	45.21	55.04	28.47	45.21
GoogleNet-GAP	52.66	28.41	47.63	55.47	30.22	47.63	55.45	30.02	47.63	50.08	26.51	47.63	55.47	30.22	47.63
GoogleNet-HAS-16	52.02	30.05	51.38	54.84	31.94	51.38	55.75	32.24	51.38	53.65	30.16	51.38	55.75	32.24	51.38

Result

	Best (GT)		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-HAS-36	54.86	26.22	42.24
AlexNet-HAS-64	55.22	27.74	44.24
AlexNet-HAS-Mix	55.04	28.47	45.21
GoogleNet-GAP	55.47	30.22	47.63
GoogleNet-HAS-16	55.75	32.24	51.38

- Best one is selected based on GT-known Loc accuracy.
- With removing multi-crop test and taking the best result among various threshold, HAS shows better performance than GAP model.
- Unlike reported at paper, Top-1 classification accuracy also increases.
- Most of best performances are acquired at threshold of 0.2 and 0.3.

Experiment 2

Dropout

Experiment

Methods	GT-known Loc	Top-1 Loc
Ours	**58.74**	**37.71**
AlexNet-dropout-trainonly	42.20	07.68
AlexNet-dropout-traintest	53.55	31.72

- Comparing with Dropout.
  - HAS works similar as dropout (more similar if patch size decreases).
    - Whereas HAS drops input while considering local structure.
  - Therefore, they compared HAS with dropped image.
    - However, I think they did not set up the experiment correctly.
      - Too much performance loss if dropping train image only while relatively alleviated even dropping test image.
      - Wierd. May be caused by distributional difference.
    - My hypothesis is that they did not normalize image before dropout.
  - Two experiments for the hypothesis.
    - 1. Input image dropout without mean normalization.
    - 2. Input image dropout with mean normalization.

Result

	Best		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-HAS-36	54.86	26.22	42.24
AlexNet-HAS-64	55.22	27.74	44.24
AlexNet-HAS-Mix	55.04	28.47	45.21
Drop-Without-Norm	50.70	0.89	1.16
Drop-With-Norm	52.85	16.15	26.90

- As expected, they might not normalize image with dropout.
- However, it shows that for input layer, hiding while considering local structure is more powerful than dropout.

Experiment Issue

$$W=\{w_1,\ldots,w_{k\times k}\}$$
$$X=\{x_1,\ldots,x_{k\times k}\}$$
$$\mu=\frac{1}{N_{pixels}}\sum_j x_j$$
$$Conv(W,X)=\begin{cases}\sum_{i=1}^{k\times k} w_i^T x_i \\ \sum_{i=1}^{k\times k} w_i^T \mu \\ \sum_{m\in visible} w_m^T x_m + \sum_{n\in hidden} w_n^T \mu \approx \sum_{i=1}^{k\times k} w_i^T \mu\end{cases}$$

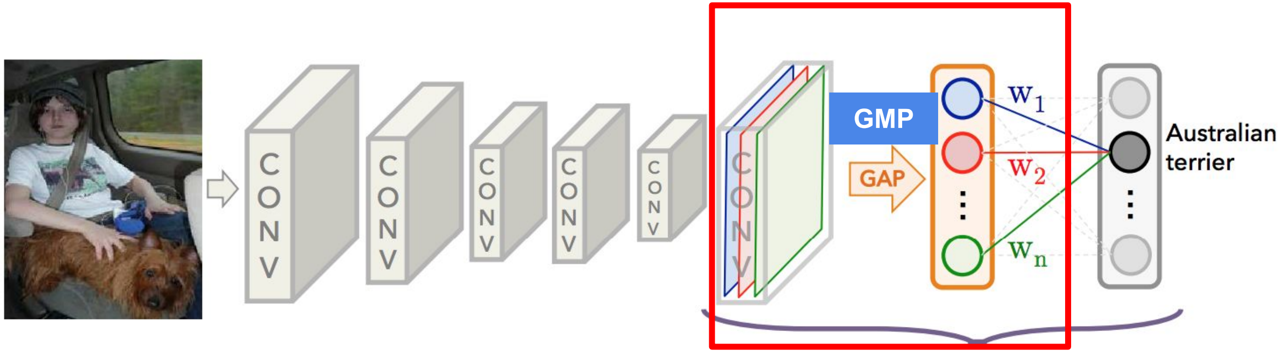
- Comparing with Dropout.
  - HAS showed better performance than dropout for input layer.
  - Can HAS be used as dropout at hidden layer?
    - Let's hiding first convolutional map.
  - However, the experiments in paper are insufficient and need more details.
    - Most important issue is that we cannot get mean (expectation) value!
      - The most important part of HAS but not mentioned :(
      - Cannot implement without it.
    - Also, to be fully compared with dropout, they should have experimented dropout too.
  - Possible implementation?
    - 1. Use moving average to approximate mean value.
    - 2. Believe batch normalization would make average to be zero and just apply dropout.

Experiment 3

Global Maximum Pooling (GMP)

Experiment

- Comparing with GMP (global max pooling).
  - In previous study, GMP showed worse performance than GAP.
  - But with HAS, GMP showed similar or even better performance than GAP.
  - To verify, I tested Alexnet-GMP and Alexnet-GMP with HAS-16.
  - Other setting are all same.



Result

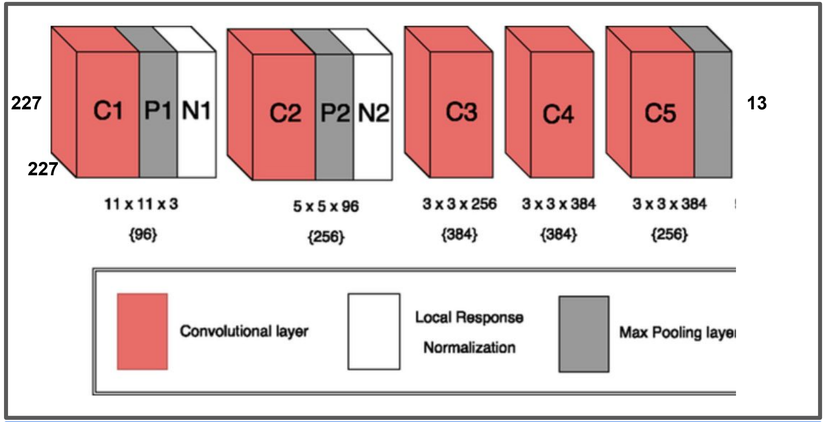
	Best		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-GMP	50.72	23.46	42.49
AlexNet-GMP-HAS-16	51.07	24.25	43.28

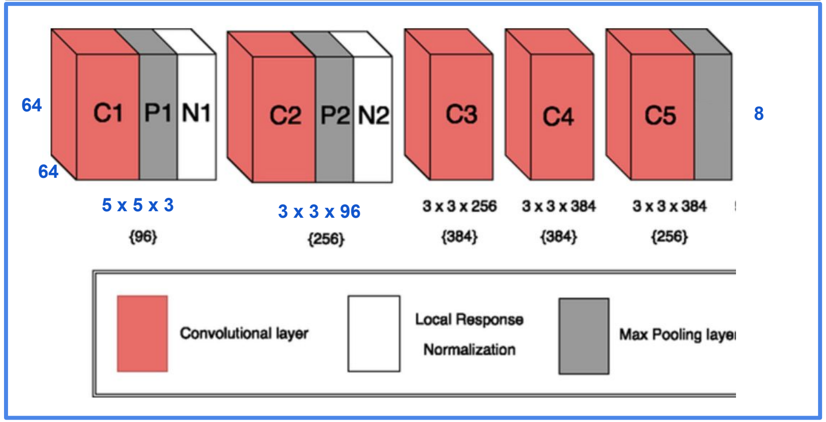
- GMP model shows better classification accuracy than GAP.
- However, in this experience, GMP does not show better localization accuracy even with HAS.

Additional Experiment 3

Resizing Image & Network

Experiment





- Input image resizing issue.
  - AlexNet and GoogleNet are originally designed for ImageNet Challenge which means that they assumes the input size is about 224x224.
  - However, our input size is 64x64.
  - Even though I naively resized them as 224 (or 227), it seems inefficient to upscale image.
- Two more experiments.
  - 1. Just put image to Alexnet without resizing resulting in 2x2 final convolution map.
  - 2. Adjust network in order to adapt small image.

Result

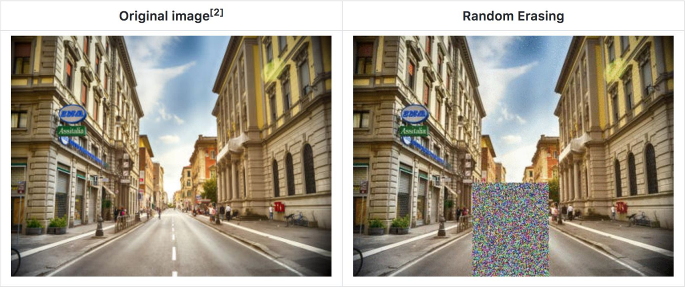
	Best		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-w/o-Resize	48.37	14.30	26.21
AlexNet-w/o-Resize-HAS-16	48.35	15.97	28.96
AlexNet-Small	53.91	23.87	38.26
AlexNet-Small-HAS-16	55.68	26.21	40.16

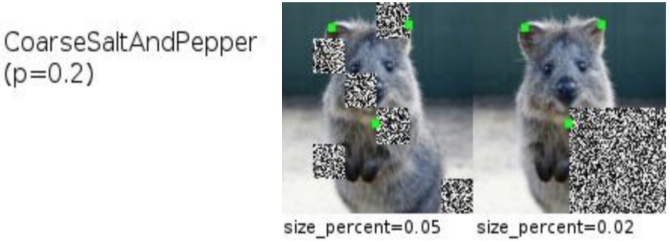
- If naive model is used without resizing, it showed very worse performance.
  - Maybe caused by too low resolution of final convolution map.
- Modified small AlexNet showed better GT-known Loc, even though total model size decreases more than 8%.
  - Can be deeper even with the same number of parameters.

Additional Experiment 4

Image Augmentation

Experiment





- HAS can be considered as image augmentation.
    - Already similar methods were exists.
    - Main difference is that HAS fill patches with mean value!
    - Multiple experiments

Methods	Flip	Rotate	Translate	Erase	Hide
AUG-1	O	O	O		
AUG-2	O	O	O	O	
AUG-1-HAS-16	O	O	O		O
AUG-2-HAS-16	O	O	O	O	O

Result

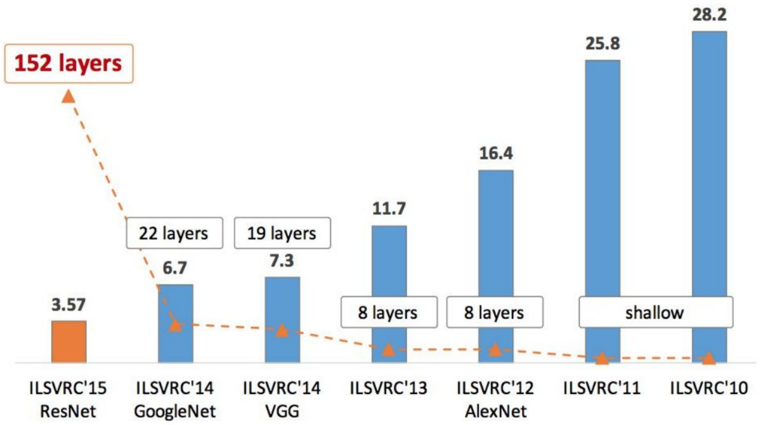
	Best		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-AUG-1	56.65	29.84	46.59
AlexNet-AUG-2	56.62	31.06	48.09
AlexNet-AUG-1-HAS-16	56.92	29.50	46.01
AlexNet-AUG-2-HAS-16	56.70	29.87	45.98

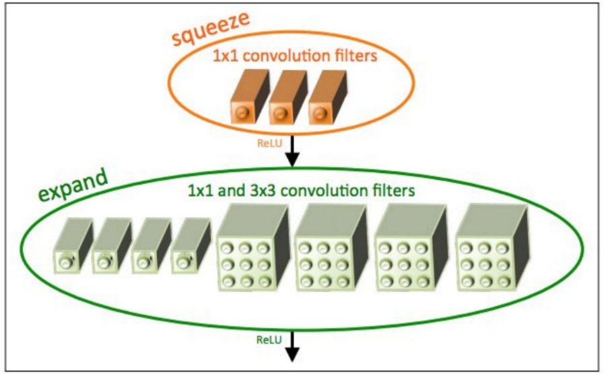
- It seems that augmentation boost the performance, both classification and localization, a lot.
- Adding HAS marginally increases localization while decreasing classification.
  - Comparing AUG-2 with AUG-1-HAS-16 shows that HAS can be improved if it is applied in stochastic way.
  - Since HAS deterministically always hides some patches.

Additional Experiment 5

Deeper Model

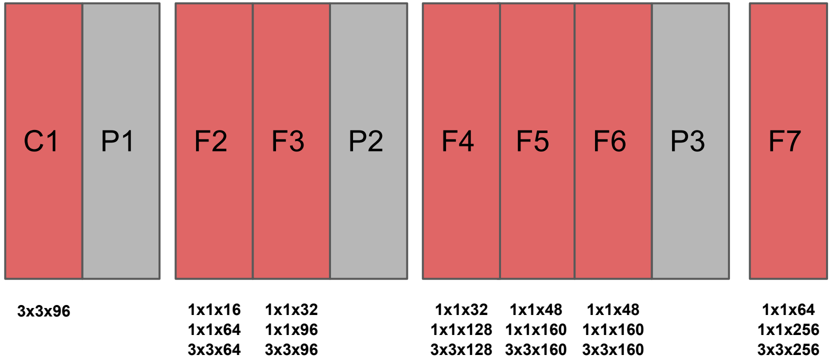
Experiment





- The deeper the better.
  - With residual connection, compression, etc.
  - Any network can be a candidates.
  - But, run out of time…
- One simple example.
  - Let's test with shallow version of SqueezeNet.
  - Main feature of it is a fire module.
    - Good compression method.

Experiment



	# Params	Cumulate	Cumulate (%)
conv1	2,688	2,688	0.09%
fire2	11,920	14,608	0.50%
fire3	35,040	49,648	1.70%
fire4	47,392	97,040	3.32%
fire5	89,456	186,496	6.37%
fire6	92,528	279,024	9.54%
fire7	184,896	463,920	15.85%
conv8	2,359,808	2,823,728	96.50%
linear	102,400	2,926,128	100.00%

- Exclude some layers of SqueezeNet.
- Do not resize input image. Rather using small window size.
- Total model size is less than 40% of AlexNet.
- Training time is between AlexNet and GoogleNet.

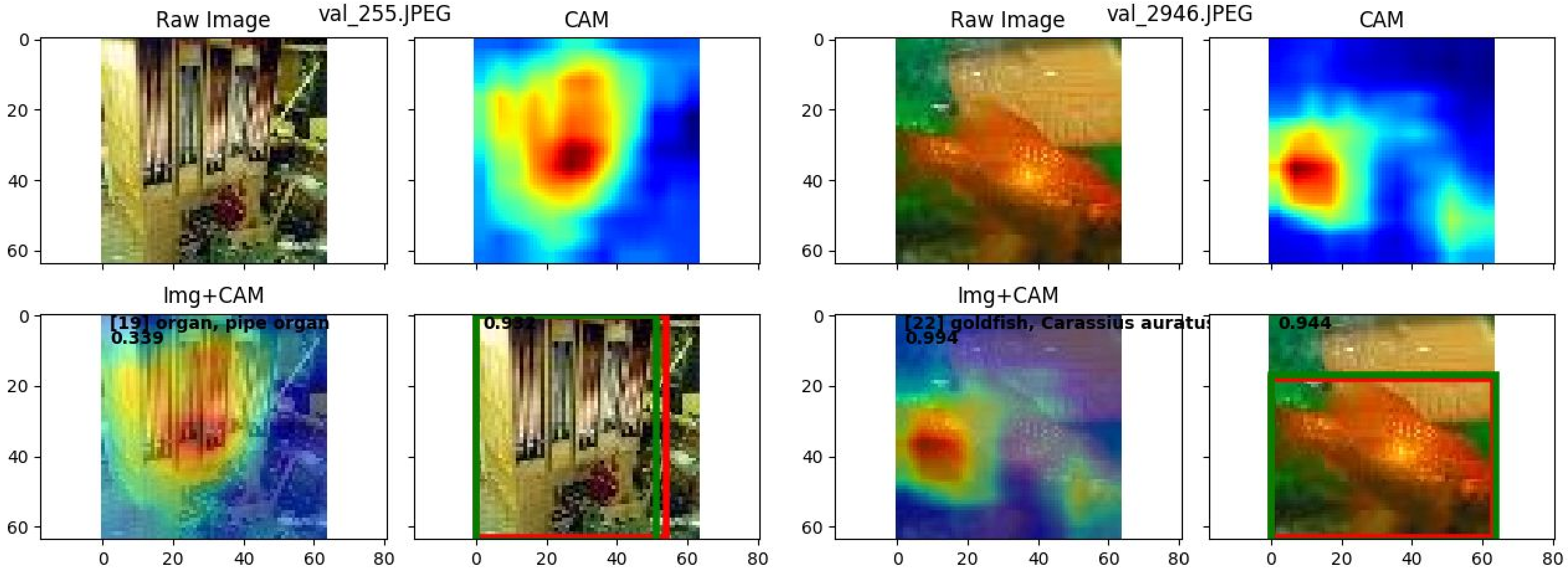
Result

	Best (GT)		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27
AlexNet-HAS-16	54.70	26.31	42.23
AlexNet-HAS-36	54.86	26.22	42.24
AlexNet-HAS-64	55.22	27.74	44.24
AlexNet-HAS-Mix	55.04	28.47	45.21
GoogleNet-GAP	55.47	30.22	47.63
GoogleNet-HAS-16	55.75	32.24	51.38
CusNet-GAP	53.35	23.29	37.89
CusNet-HAS-16	55.06	28.74	45.80
CusNet-HAS-36	55.41	29.68	47.80
CusNet-HAS-64	54.95	29.18	47.01
CusNet-HAS-Mix	55.68	31.20	49.70

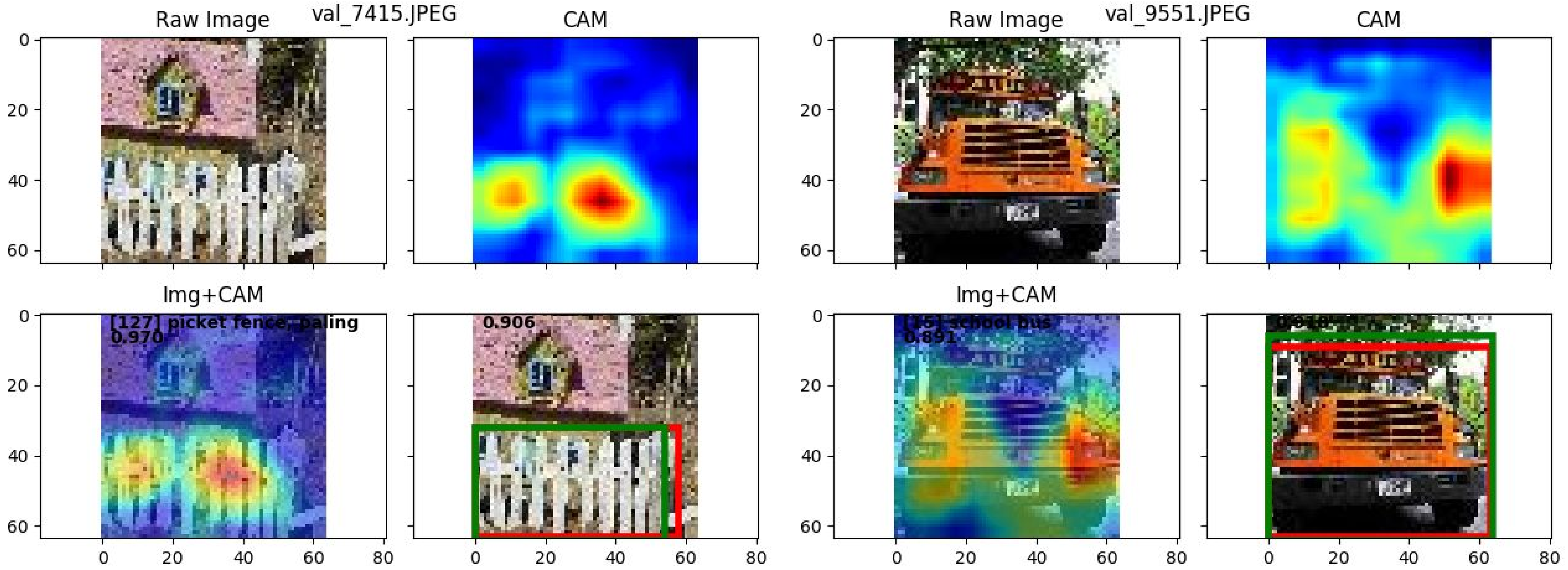
- Even with smaller number of parameters, using fire module showed similar or better performance.
- If using deeper and complexer model like modern InceptionNet or Resnet, all metrics might show better scores.
- Replacing last convolution layer can reduce more parameters.

Discussion

Visualization



Visualization



Result Table

	Best				Best		
Methods	GT-known Loc	Top-1 Loc	Top-1 Clas	Methods	GT-known Loc	Top-1 Loc	Top-1 Clas
AlexNet-GAP	54.31	24.53	39.27	Drop-Without-Norm	50.70	0.89	1.16
AlexNet-HAS-16	54.70	26.31	42.23	Drop-With-Norm	52.85	16.15	26.90
AlexNet-HAS-36	54.86	26.22	42.24	AlexNet-GMP	50.72	23.46	42.49
AlexNet-HAS-64	55.22	27.74	44.24	AlexNet-GMP-HAS-16	51.07	24.25	43.28
AlexNet-HAS-Mix	55.04	28.47	45.21	AlexNet-w/o-Resize	48.37	14.30	26.21
GoogleNet-GAP	55.47	30.22	47.63	AlexNet-w/o-Resize-HAS-16	48.35	15.97	28.96
GoogleNet-HAS-16	55.75	**32.24**	**51.38**	AlexNet-Small	53.91	23.87	38.26
CusNet-GAP	53.35	23.29	37.89	AlexNet-Small-HAS-16	55.68	26.21	40.16
CusNet-HAS-16	55.06	28.74	45.80	AlexNet-AUG-1	56.65	29.84	46.59
CusNet-HAS-36	55.41	29.68	47.80	AlexNet-AUG-2	**56.62**	**31.06**	**48.09**
CusNet-HAS-64	54.95	29.18	47.01	AlexNet-AUG-1-HAS-16	**56.92**	29.50	46.01
CusNet-HAS-Mix	55.68	**31.20**	**49.70**	AlexNet-AUG-2-HAS-16	**56.70**	29.87	45.98

Discussion

- HAS generally increases localization performance.
- In my case, classification accuracy also increases.
- Using good classification model makes localization more powerful.
- By the way, among all experiment, GT-known localization score does not vary a lot.
    - May be problem of dataset.
    - Bounding boxes are usually too big since images of Tiny Imagenet is cropped.
- Low accuracy of dataset may also derived from dataset issue.
    - Where Tiny ImageNet comes from?
    - Stanford cs231 course.
    - Reports of that course said that accuracy of around 40% is normal case.
    - Train: 456,567 -> 100,000
    - Valid: 20,121 -> 10,000

Discussion

- HAS works but augmentation also works (even better).
  - Less impressive than before I tried augmentation.
- Future works?
  - Ensemble test.
  - Stochastic HAS.
  - Replace last convolution layer of CAM with compressed version.
  - Multiscale CAM.
    - Current CAM is hard to detect relatively small object.
    - Multiscale setup might be helpful.

Thank you.