SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE

1. Introduction & Motivation

동일한 정확도에서 더 가벼운 CNN의 장점은?

- 효율적인 distributed training

  distributed data-parallel training에 있어서 communication overhead(통신하는데에 들어가는 간접적인 시간)는 parameter의 수에 비례한다. 즉, 가벼운 모델은 통신량이 줄어들기 때문에 더 빨리 훈련이 가능하다.

- 실시간으로 정보를 전송해야하는 업무에 적합

  예를 들어, 자율 주행 시스템은 계속 바뀌는 정보를 빠르게 처리하기 위해 주기적으로 차량 정보에 서버를 업데이트한다. 모델의 연산량이 적으면, 정보 갱신을 더 빈번하게 할 수 있다.

- FPGA(10MB로 용량이 제한되어있는 메모리)와 임베디드 시스템에 적용 가능

  모델의 용량이 작다면 메모리가 제한되어 있는 분야에서도 딥러닝을 적용할 수 있다.

논문의 모델을 구성하게된 동기는?

위 장점들을 바탕으로 CNN architecture에서 parameter의 수는 낮추면서 정확도는 유지시키는 방법에 집중하게 되었고, 그 모델이 SqueezeNet이다.

2. SqueezeNet : preserving accuracy with few parameters

- Architectural Design Strategies

  논문 저자는 파라미터 수는 적지만 정확도는 어느정도 유지하는 CNN 구조를 구축하기 위해 3가지 전략을 이용한다. 아래 **3가지 전략을 이용하여 Fire module이 탄생**한다.

- **Strategy 1. 3x3 filter를 1x1 filter로 대체한다.**

  1x1 filter는 3x3 filter 연산량보다 9배 적다.

- **Strategy 2. 3x3 filter에 입력되는 채널의 수를 줄인다.**

  오로지 3x3 filter로 구성된 Conv layer가 있다고 해보자. 연산량은 (input channel)*(filter 수=output channel)*(3*3)이다. 입력 채널의 수를 감소시키면 3x3 filter의 연산량이 감소한다.

- **Strategy 3. Conv layer가 큰 activation map을 갖도록 Downsample을 늦게 수행한다.**

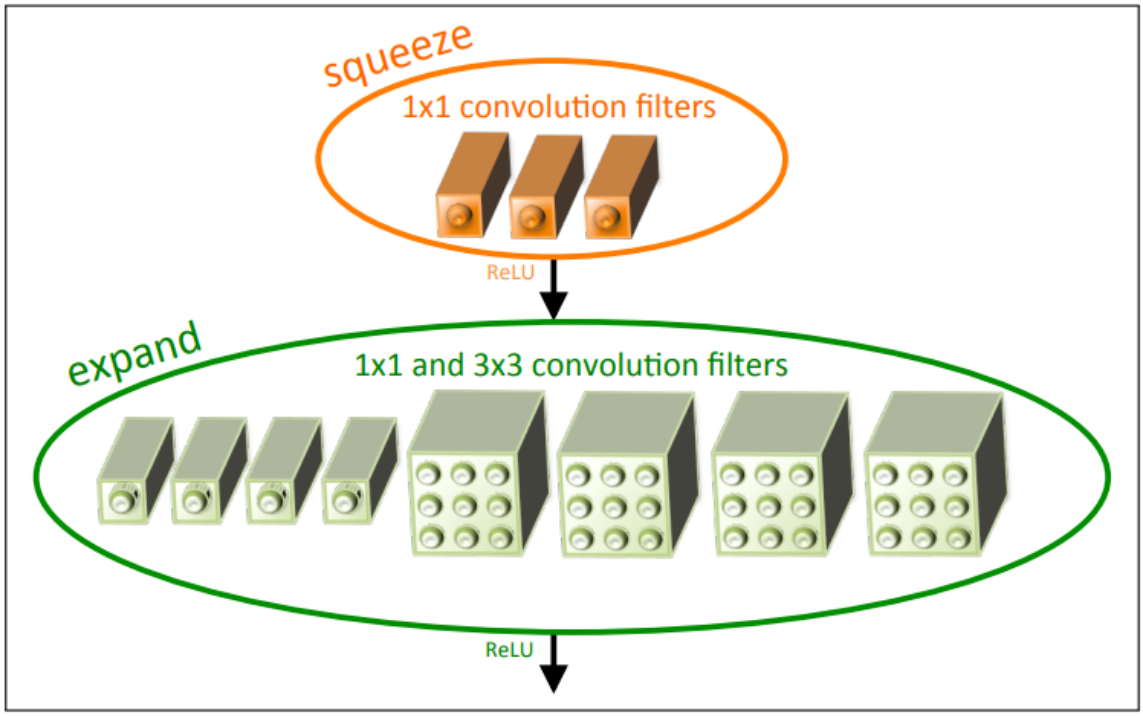
  - Downsampling : stride가 1 이상인 Convolution 또는 Pooling

  큰 activation을 가지고 있을수록 정보 압축에 의한 손실이 적어 성능이 높다.

  따라서, 정보 손실을 줄이기 위해 네트워크 후반부에 downsampling을 수행한다.

전략 1,2는 정확도를 유지하면서 parameter 수를 줄이기 위함이었고, 전략 3은 제한된 parameter에서 정확도를 최대화하기 위함이다.

- The Fire Module



FIre Module은 총 두 가지 Layer로 이루어져 있다.

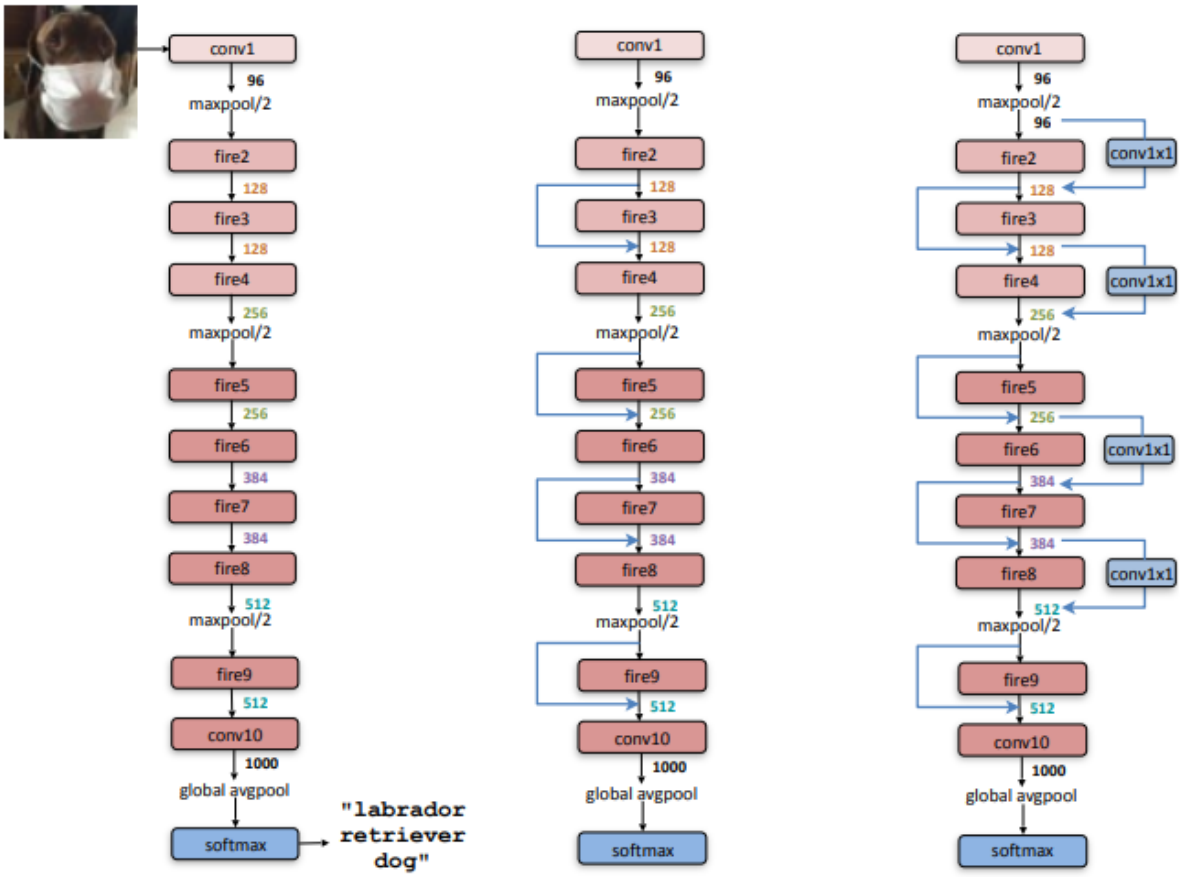
- **Squeeze Convolution Layer** : 1x1 filter로만 구성
- **Expand Convolution Layer** : 1x1 filter와 3x3 filter로 구성

Fire Moduel에는 총 세 가지 hyperparameter가 있다.

- s_{1x1} : squeeze layer에서의 1x1 filter의 총 개수
- e_{1x1} : expand layer에서의 1x1 filter의 총 개수
- e_{3x3} : expand layer에서의 3x3 filter의 총 개수

Fire Module을 만들 때, $s_{1x1} < e_{1x1} + e_{3x3}$로 설정하여 **squeeze layer의 channel 수가 expand layer의 channel 수보다 작게 설정**한다. squeeze layer의 출력값은 expand layer의 입력값으로 생각할 수 있다. 입력값을 expand layer의 filter보다 작게 설정하여 2번 전략을 만족한다.

- The SqueezeNet Architecture



1. SqueezeNet : 1개의 Conv Layer → 9개의 Fire Module → 1개의 Conv Layer → Softmax

   Conv1, FIre4, Fire8 이후에 MaxPooling(Stride=2) Layer를 수행하여 해상도를 줄여나갔고, Conv10 뒤에는 Avg Pooling Layer를 통해 output size를 조정하였다. (MaxPooling의 위치는 3번 전략과 관련있다고 하는데 잘 모르겠다. 그냥 적게 썻다는 표현이 더 맞는것 같다.)

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations		parameters				compression info				1,248,424 (total)	**421,098** (total)

- 1x1 filter와 3x3 filter의 output activation이 같은 height와 width를 갖기 때문에 expand module의 3x3 filter에 입력되는 데이터에 1-pixel 짜리 zero padding을 추가해준다.
- Squeeze Layer와 Expand Layer에 ReLU activation 적용
- Fire9 이후 Dropout with p=0.5 적용
- FC layer가 부족하여 NiN(Network in Network) 구조 활용
  - Squeeze layer 안에 fire module이 있는 구조
- 초기 학습률 0.04로 설정 후 점차 감소
- Caffe 프레임워크는 Multiple filter resolution이 있는 Convolution을 지원해주지 않아 1x1 filter와 3x3 filter를 나눠서 구현하여 concate하는 식으로 구현

3. Evaluation of SqueezeNet

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

다양한 Model Compression 기법과의 비교이다.

단순 SqueezeNet만 사용했을 때는 50배 가까이 모델 사이즈가 줄어들었다. 게다가 기존 AlexNet의 top-1 & top-5 accuray에 근접하거나 뛰어넘는 모습을 보여준다.

여기에 더해 uncompressed 된 32bit의 데이터 타입을 사용한 생짜 SqueezeNet과 deep compression을 적용한 8bit, 6bit짜리 데이터 타입을 사용한 결과, 최고의 결과물은 모델 사이즈가 510배까지 줄어들었으며 성능도 큰 차이가 나지 않는다. 이는 SqueezeNet 또한 모델 압축에 굉장히 유연하다는 뜻이다.

따라서 **AlexNet에 model compression기법을 적용한 결과보다 새롭게 제안한 SqueezeNet이 경량화 되었지만 더 좋은 성능을 보여준다.**
그리고 **Deep compression과 같은 압축기법이 이미 경량화된 구조(SqueezeNet)에서도 효과가 있음을 증명**한다.

4. CNN Micro Architecture 설계공간 탐색

microarchitecture 탐색과 macroarchitecture 탐색을 진행해보겠다.

microarchitecture는 fire module과 같은 하나의 block를 의미하며, macroarchitecture는 SqueezeNet과 같은 전체 network를 의미한다.

- Metaparameter 설정

  각각의 Fire Module에는 $s_{1x1}, e_{1x1}, e_{3x3}$ 3 dimensional hyperparameter가 필요로 하다. 따라서SqueezeNetd의 8-layer FireModules는 24 dimensional hyperparameter가 필요하다.

SqueezeNet에서는 설계전략에 맞는 network 설정을 위해 24개의 hyperparameter를 통솔하는 metaparameter가 필요하다.

- $base_e$: 첫번째 FIre Module의 expand filter의 수
- $freq$: fire module의 셋팅이 바뀌는 주기
- $incr_e$: 매 $freq$마다 expand fitler의 증가율
- $e_i = base_e + incr_e * \frac{i}{freq} = e_{i,1x1} + e_{i,3x3}$: i번째 expand filter의 수
- $pct_{3x3} \in [0, 1]$: expand filter 중 3x3 filter 수의 비율
- $SR(\text{squeeze ratio})$: expand layer 앞에 있는 squeeze layer의 filter 수로 expand layer의 filter에 대한 비율

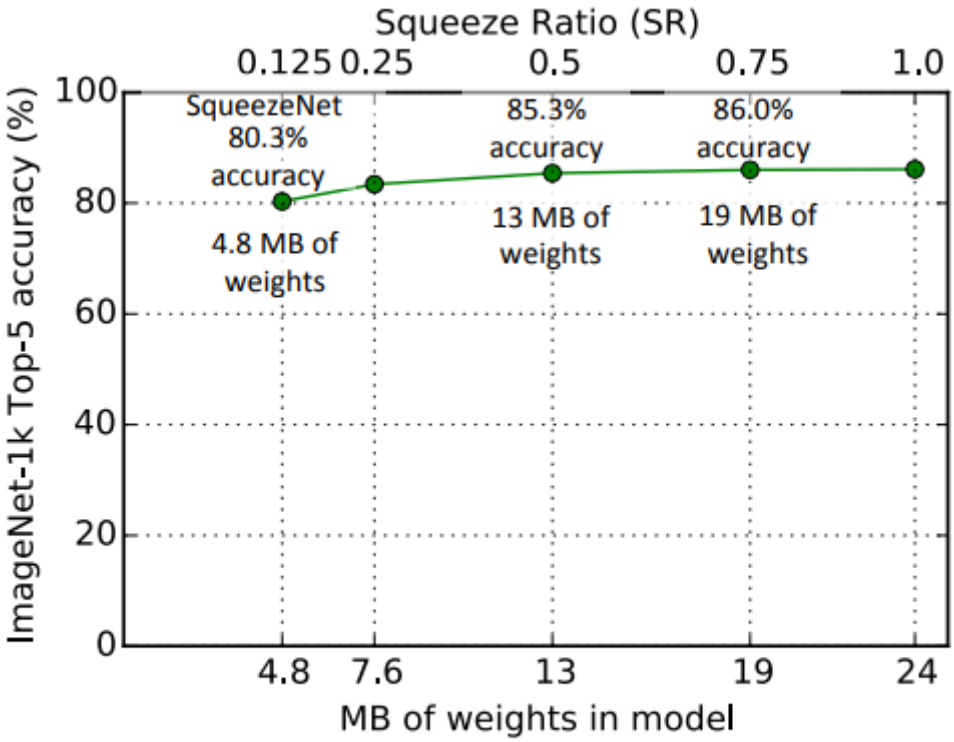
따라서, $e_{i,3x3} = pct_{3x3} \times e_i$와 $s_{i,1x1} = SR \times e_i$가 된다.

- Squeeze Ratio에 따른 성능 변화

  설정한 metaparameter를 바탕으로 $SR(\text{squeeze ratio})$을 조절해가며 모델 크기에 따른 성능을 실험을 통해 확인

metaparameter	value
$base_e$	128
$incr_e$	128
pct_{3x3}	0.5
$freq$	2
SR	[0.125, 1.0]

  실험 결과는 SR 0.125부터 0.75까지는 모델의 크기와 성능이 모두 증가한다. 하지만 SR이 0.75를 넘어가는 순간부터는 성능 향상 없이 모델의 크기만 증가한다.



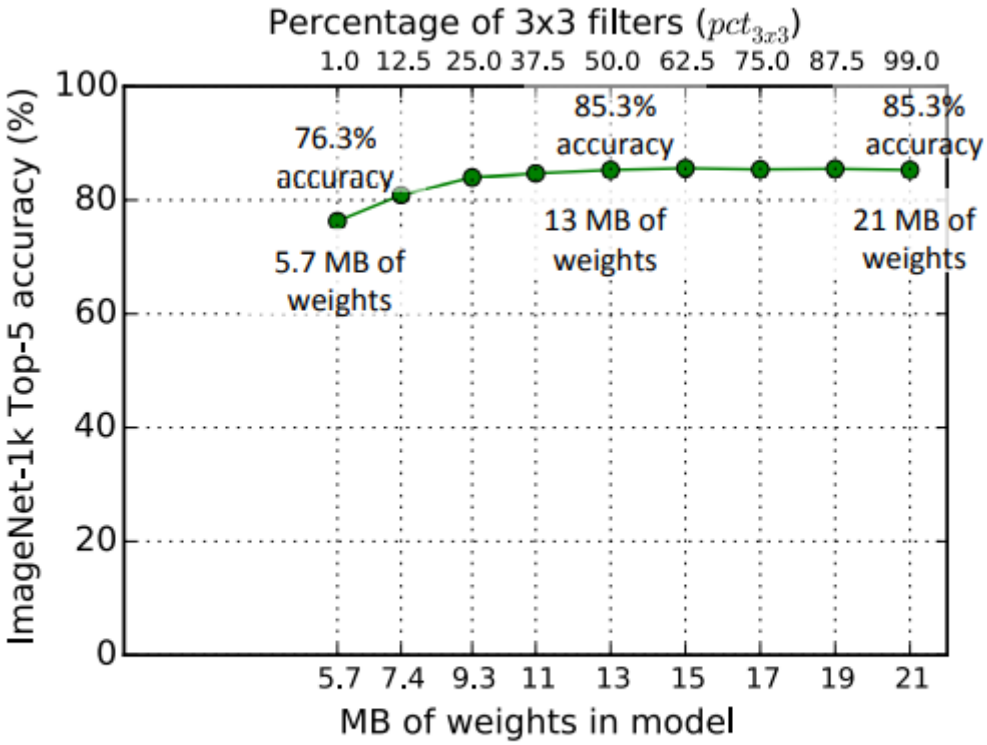
(a) Exploring the impact of the squeeze ratio (SR) on model size and accuracy.

- 1x1 / 3x3 비율에 따른 성능 변화

  이번 실험은 expand filter 중 1x1 filter와 3x3 filter의 비율에 따른 성능 변화로 filter의 **공간적 해상도가 얼마나 중요한지**를 살펴보는 것이다.

metaparameter	value
$base_e$	128
$incr_e$	128
SR	0.5
$freq$	2
pct_{3x3}	[0.01,0.99]

  실험 결과는 pct_{3x3}이 50%까지는 성능이 향상되지만 50%를 초과하는 순간부터는 성능 향상 없이 모델의 용량만 증가한다.



(b) Exploring the impact of the ratio of 3x3 filters in expand layers (pct_{3x3}) on model size and accuracy.

5. CNN Macro Architecture 설계공간 탐색

FIre module을 활용해 어떤 조합을 해야 최적의 성능이 나오는지 실험을 통해 밝혀냈다.

실험은 ResNet으로부터의 영감을 받아서 Fire module 간의 high-level connection에 따른 성능 변화를 살펴보았다.

실험에 사용하는 모델은 크게 세 종류가 있다.

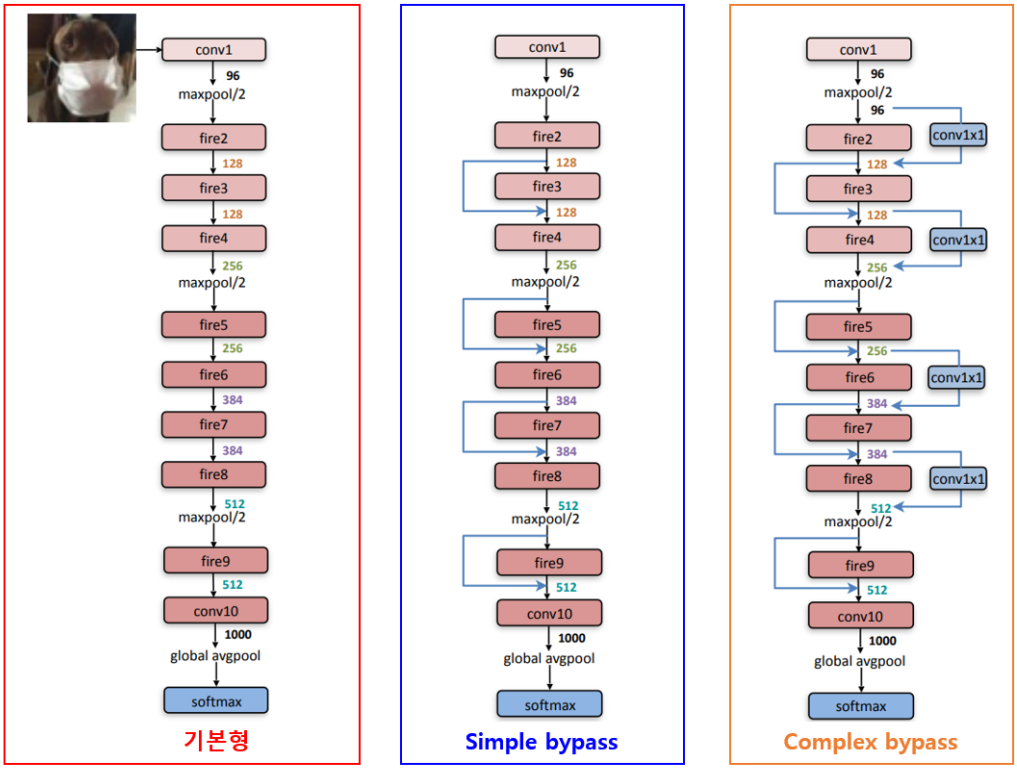


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

- 기본형 Vanilla SqueezeNet
- SqueezeNet with simple bypass(resnet의 skip connection과 동일) connections between some Fire modules
  - FIre4의 input을 element 간의 덧셈인 Fire2의 output+Fire3의 output으로 설정
  - 추가적인 파라미터를 필요가 없다
- SqueezeNet with complex bypass connections between the remaining Fire modules
  - Simple Bypass 모델에서 채널수가 달라지는 영역에 1x1 Conv layer를 추가한 형태
  - 추가적인 파라미터를 필요로 한다.

SqueezeNet의 Fire module은 squeeze layer에서 채널 수가 감소하는 bottle neck 구조가 된다. 예를 들어 $SR = 0.125$일 때, squeeze layer의 채널 수는 expand layer 채널 수의 $\frac{1}{8}$이다. 따라서 채널 감소에 따른 정보 손실이 발생한다. 하지만 Bypass를 추가하면 skip-connection을 통해 손실된 정보가 보전되기 때문에 성능이 향상될 것을 기대할 수 있다.

실험 결과에서 Simple Bypass와 Complex bypass를 추가했을 때 모두 성능이 향상되었다. 하지만 Complex Bypass보다 Simple Bypass의 성능이 더 좋았다. 따라서 Simple Bypass 모델이 모델 사이즈 증가 없이 성능을 향상시킬 수 있는 좋은 방법이라는 것을 알 수 있다.

Architecture	Top-1 Accuracy	Top-5 Accuracy	Model Size
Vanilla SqueezeNet	57.5%	80.3%	4.8MB
SqueezeNet + Simple Bypass	**60.4%**	**82.5%**	4.8MB
SqueezeNet + Complex Bypass	58.8%	82.0%	7.7MB