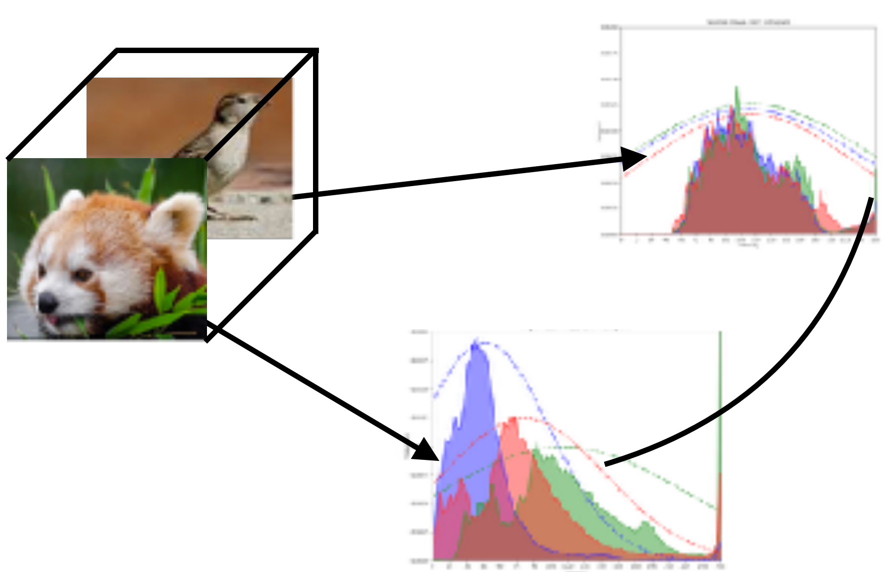
Batch Normalization

Why normalization need?



Problem

Different pixel distribution make hard train image features.
To avoid , gradient vanishing , gradient exploding , set learning rate to low

Solution

Normlization
Normlization

$$\frac{x - x_{min}}{x_{max} - x_{max}}$$

Batch normalization

Batch normalization

STEP 1

$$x'_{each\_pixel} = \frac{x_{each\_pixel} - E(x)_{batch}}{var(x)_{batch}}$$

STEP 2

$$X = \gamma * x + \beta$$

Neural network 을 학습

"Note that simply normalizing
each input of a layer may change what the layer can represent.
For instance, normalizing the inputs
of a sigmoid would constrain them to the linear regime of the nonlinearity."

STEP 3

$$\frac{\Sigma_i E(x)_{i,batch}}{K} = E(x)_{test}$$

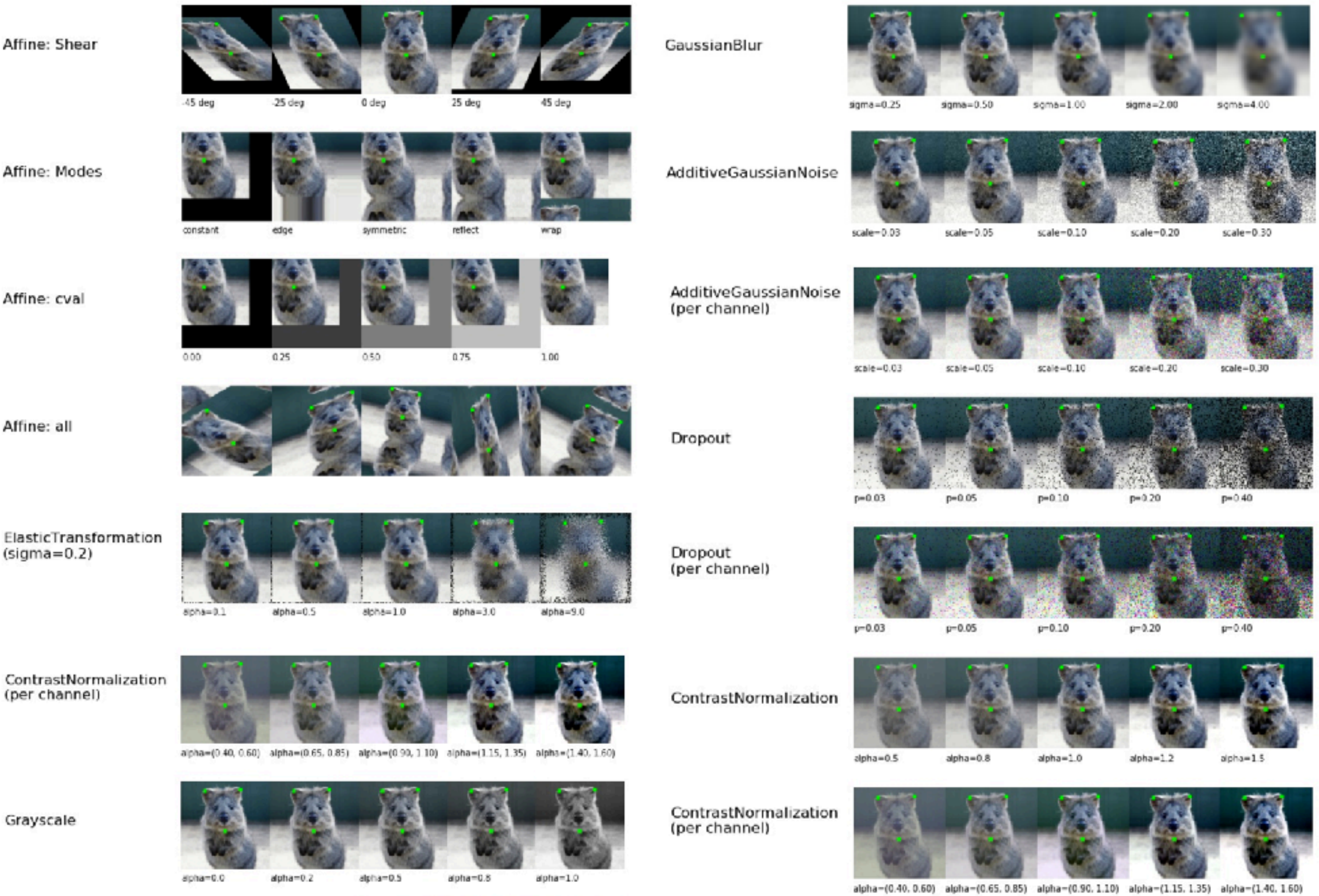
$$\frac{\Sigma_i var(x)_{i,batch}}{K} = var(x)_{test}$$

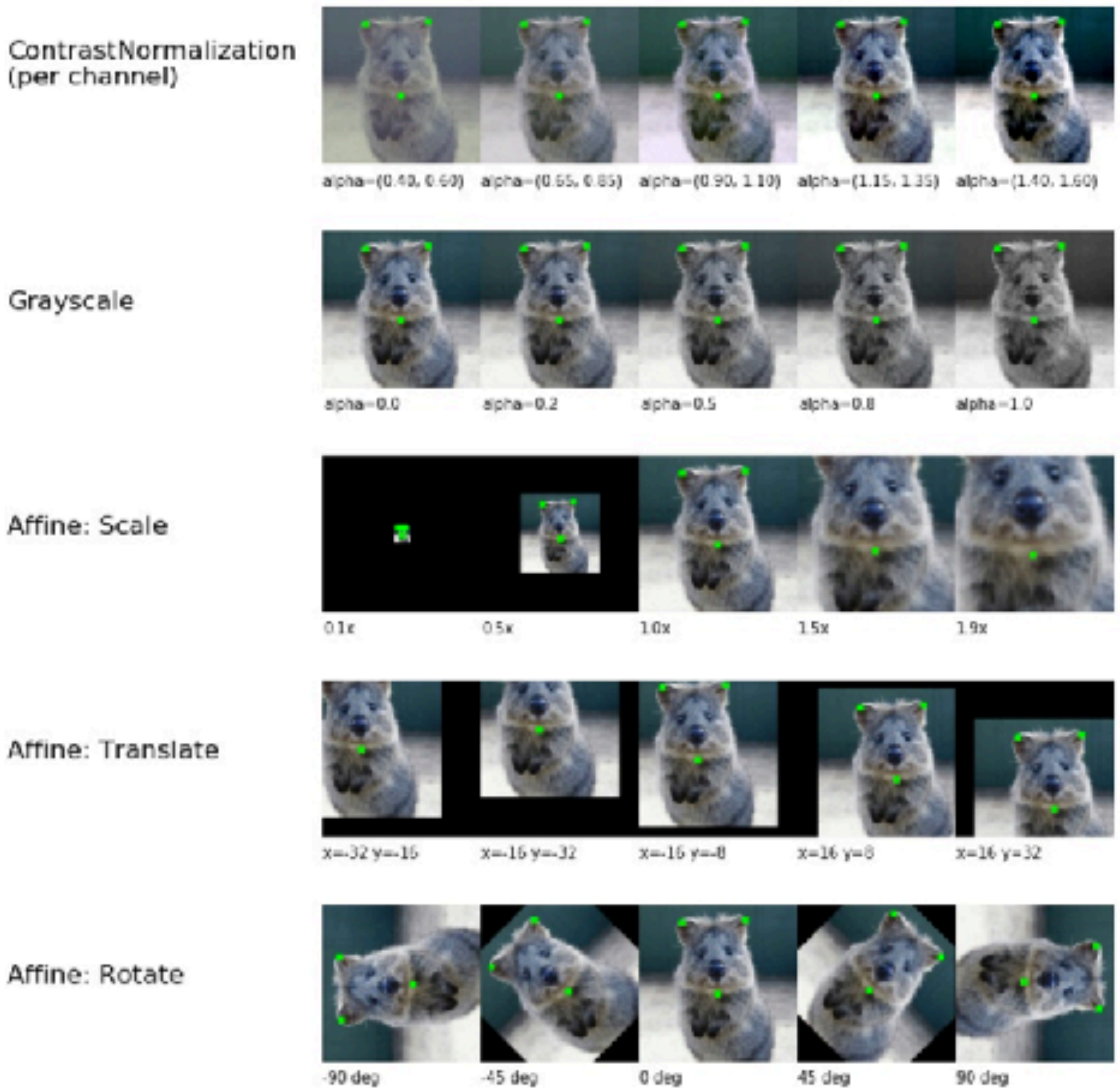
**Test 학습을 할때 마다 mean. , var 이 달라지기 때문에
변하지 않는 mean , var 이 필요하다.**

성능 평가

```
import tensorflow as tf
x = tf.constant(10.0, name='input')
w = tf.Variable(0.8, name='weight')
y = tf.multiply(w, x, name='output')
y_ = tf.constant(0.0, name='correct_value')
loss = tf.pow(y - y_, 2, name='loss')
train_step = tf.train.GradientDescentOptimizer(0.025).minimize(loss)

for value in [x, w, y, y_, loss]:
    tf.summary.scalar(value.op.name, value)

summaries = tf.summary.merge_all()

sess = tf.Session()
summary_writer = tf.summary.FileWriter('log_simple_stats', sess.graph)

sess.run(tf.initialize_all_variables())
for i in range(100):
    if i % 10 ==0:
        print("epoch {}, output: {}".format(i, sess.run(y)))
    summary_writer.add_summary(sess.run(summaries), i)
    sess.run(train_step)
```

AUGMENTATION




ContrastNormalization (per channel)
alpha=(0.40, 0.60)
alpha=(0.65, 0.85)
alpha=(0.90, 1.10)
alpha=(1.15, 1.35)
alpha=(1.40, 1.60)
Grayscale
alpha=0.0
alpha=0.2
alpha=0.5
alpha=0.8
alpha=1.0
Affine: Scale
0.1x
0.5x
1.0x
1.5x
1.9x
Affine: Translate
x=-32 y=-16
x=-16 y=-32
x=-16 y=-8
x=16 y=8
x=16 y=32
Affine: Rotate
-90 deg
-45 deg
0 deg
45 deg
90 deg

Gradient Optimization

http://sebastianruder.com/optimizing-gradient-descent/

모든 파라미터 Θ:parameter **을 최소화 하기 위해**

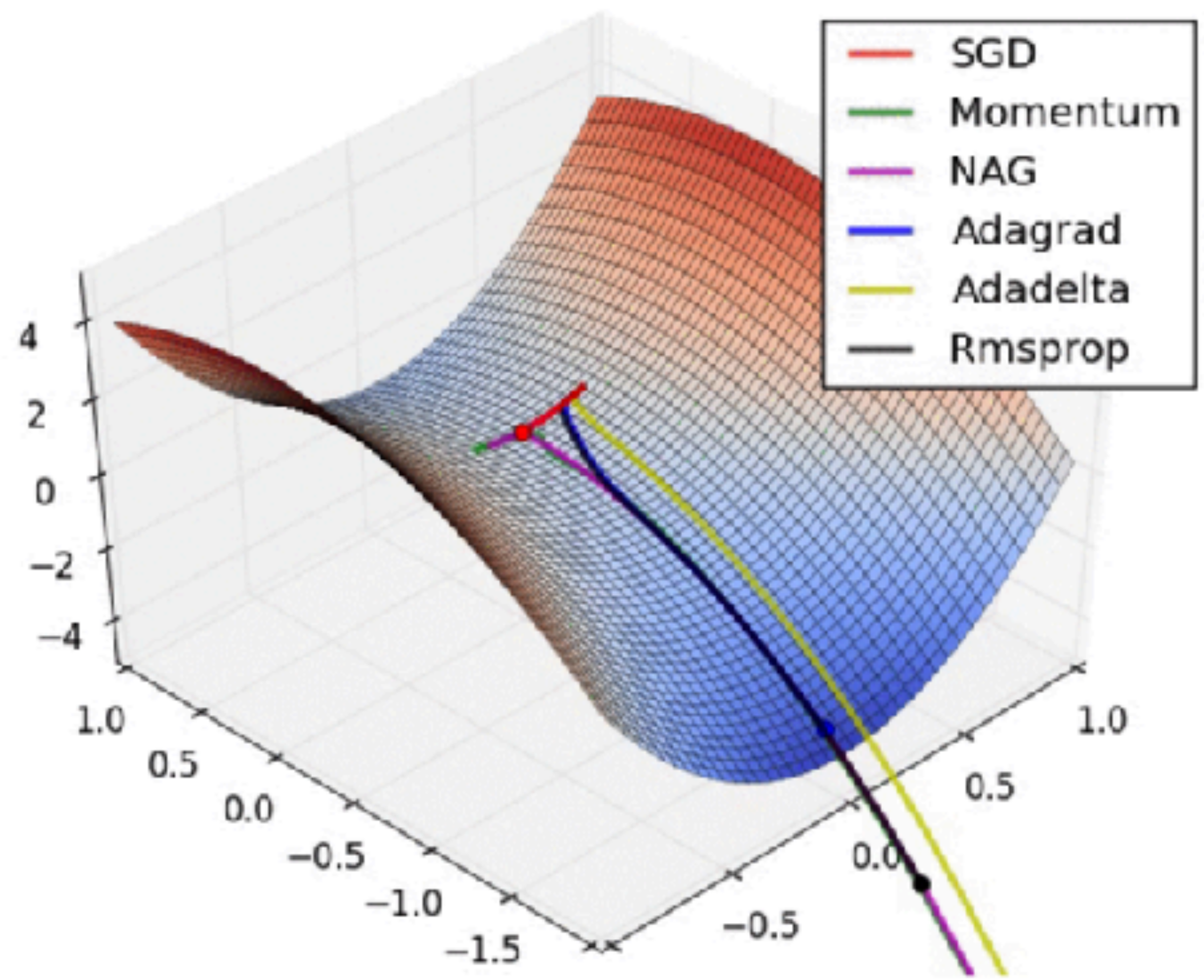
J(θ):Loss function **을 줄이기 위해 .**

$\nabla_\theta J(\theta)$ **을 이용한다.**

어떻게?

기울기의 반대방향으로

$\theta = \theta - \eta \nabla_\theta J(\theta)$



Gradient Descent Optimization Algorithms at Long Valley

Optimizer

Momemtum

SGD(Stochastic Gradient Descent (SGD))

NAG(Nesterov Accelerated Gradient)

Adagrad

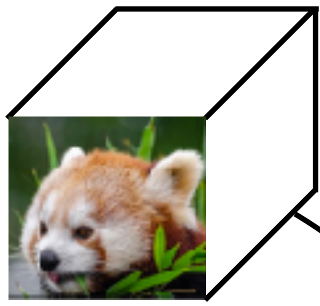
RMSProp

AdaDelta

Adam

SGD

50000



BATCH DESCENT GRADIENT

단점.
이동속도가 현저하게 느리다
Local minima에 쉽게 빠진다

한장을 꺼내서 한번 학습하고 ,
차례대로 50000장의 이미지를 모두 꺼내 학습힌다

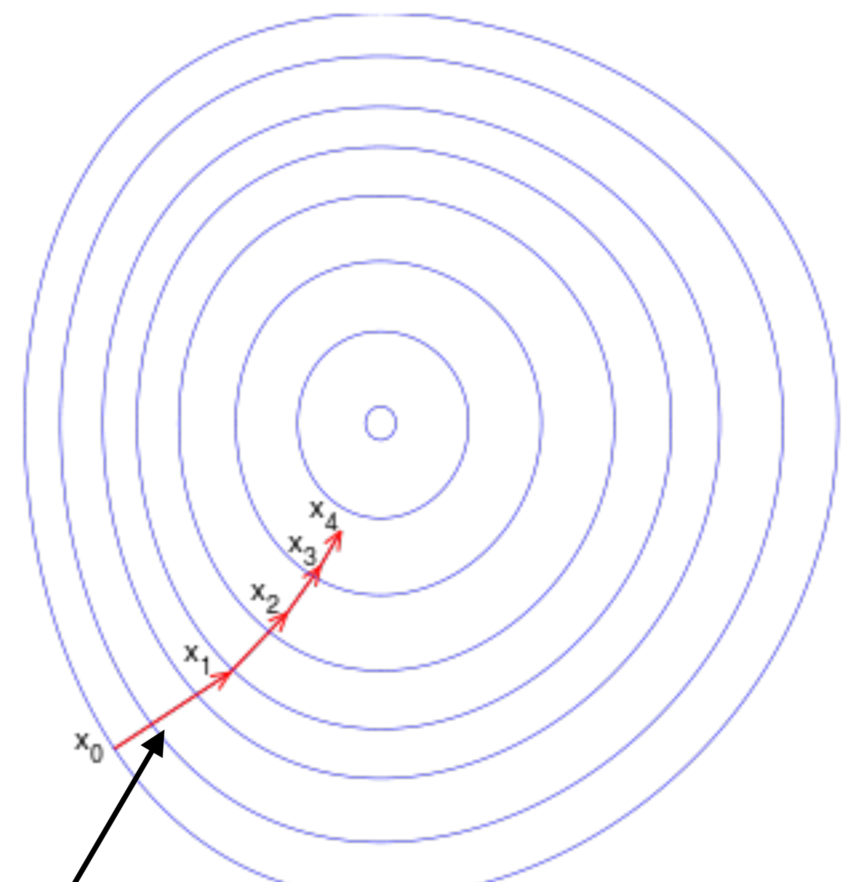
$\theta = \theta - \eta \nabla_\theta J(\theta)$

64

η:learning rate

$\theta_{batch} = \theta_{batch} - \eta \nabla_\theta J(\theta)_{batch}$

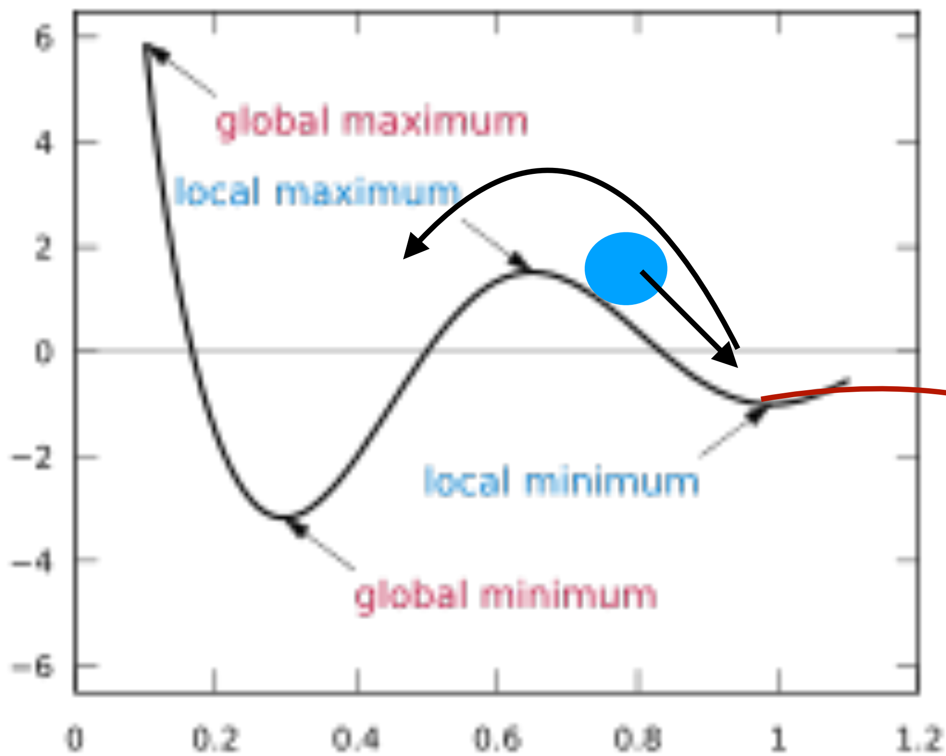
$[J(\theta)_{batch} = \Sigma J(\theta)_{batch} / batch]$



등고선이 크기(기울기) 일정하다 라고 생각하기 때문에
SGD는 optimization 하는데 속도가 느리고 한계점이 있다.

그러면 기울기에 따라 변화량(∇)에 따라 learning rate 을 변하게 하면 안될까?

Local minima



$$\theta_{batch} = \theta_{batch} - \eta \nabla_{\theta} J(\theta)_{batch}$$

↓

$$\nabla_{\theta} J(\theta)_{batch} = 0$$

↓

학습 종료

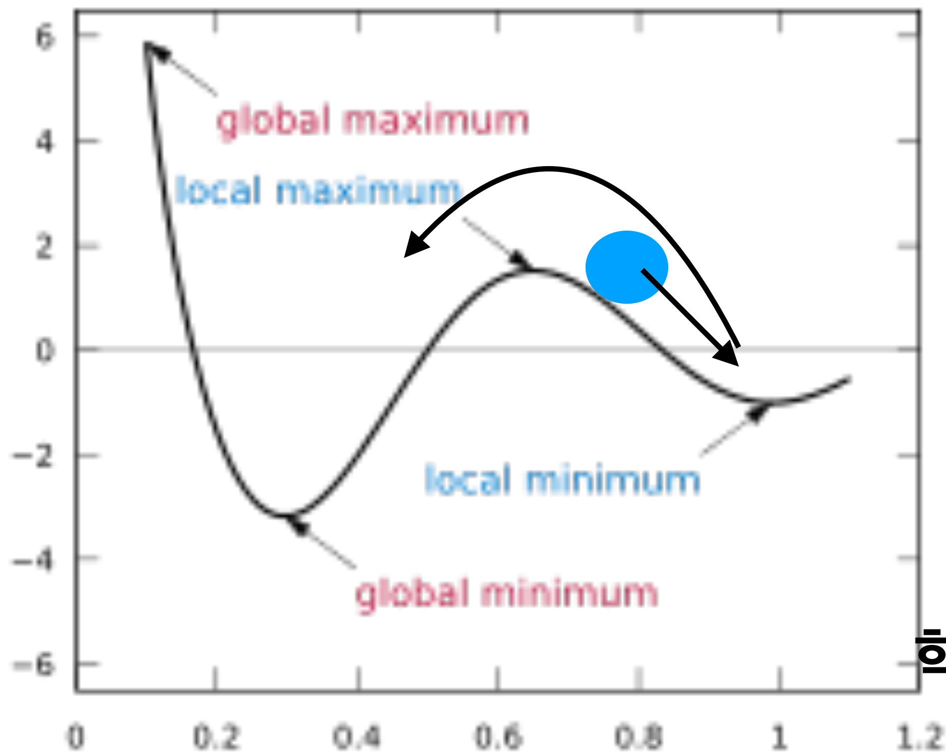
Momentum

Learning rate 을 고정하지 말고 변화량에 따라 learning rate 을 변화시키자!

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

$\gamma : momentum$

$$v_t = \eta \nabla_\theta J(\theta)_t + \gamma \eta \nabla_\theta J(\theta)_{t-1} + \gamma_2 \eta \nabla_\theta J(\theta)_{t-2} + \ldots$$



**이 공이 내려올때 더 큰 가중치를 받고 올라가면
Local minima을 뛰어 넘을수 있지 않을까?**

학습 속도가 느리다.

https://github.com/SoulDuck/resnet

Momentum

학습 속도가 느리다.

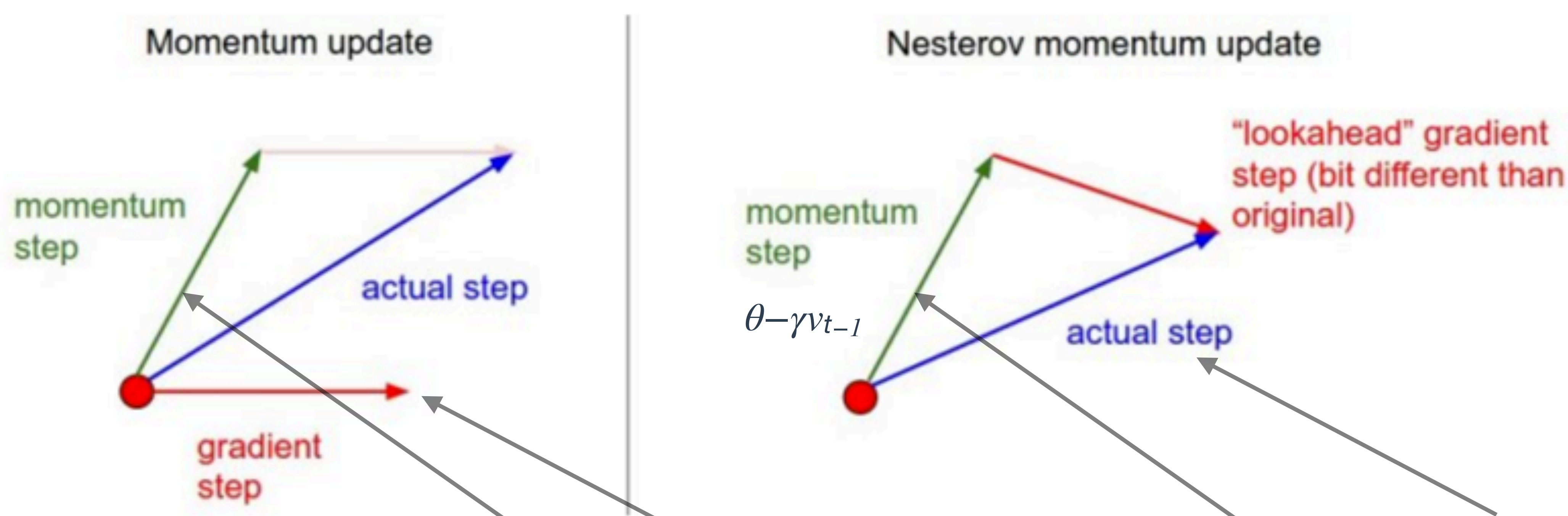
변수를 저장해야 한다. $Because.. .v_t=\eta\nabla_\theta J(\theta)_t+\gamma\eta\nabla_\theta J(\theta)_{t-1}+\gamma_2\eta\nabla_\theta J(\theta)_{t-2}+\dots$

Global minimum을 빠져나간다.

↓

멈출때 관성텀을 적게 줘서 멈추게 하는 방법이 있을까?

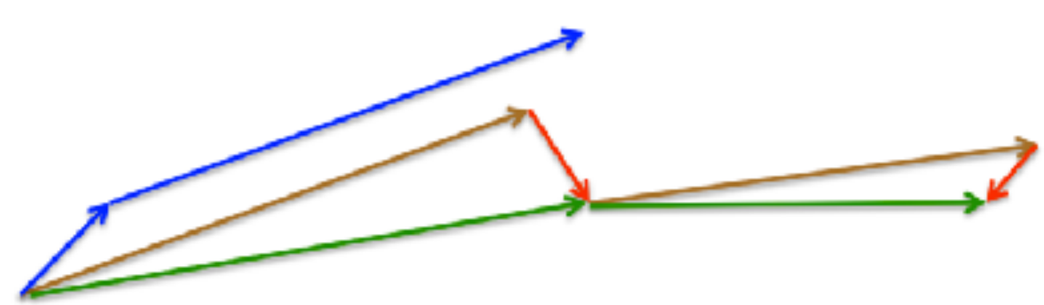
Nesterov Accelerated Gradient



$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$

$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$

momentum이 공이 흘러내리는 방향과 다른쪽의 힘을 준다...

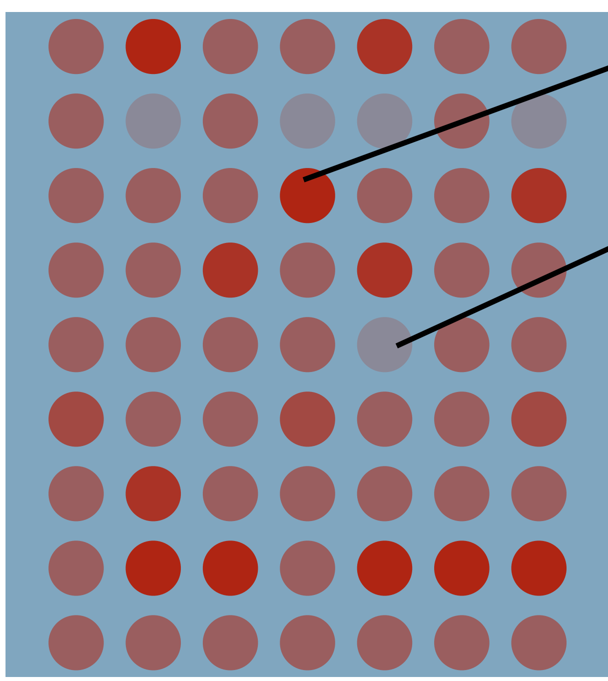


이미지 4 : 네스테로프 업데이트 (출처 : G. Hinton 강의 Sc)

Adagrad (Adaptive gradient)

Adaptive:

이미 많이 변화한 변수들은 변화량을 적게하고 적게 변화한 변수들은 변화량을 크게하자



변화량이 큰 변수 G_{18}

step size decay

변화량이 작은 변수 G_{33}

$$G_t = G_{t-1} + (\nabla_\theta J(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot \nabla_\theta J(\theta_t)$$

step size을 신경쓰지 않는다.

```
for i in range(10000):
    msg = '\r Progress {0}/{1}'.format(i,10000)
    sys.stdout.write(msg)
    sys.stdout.flush()
    if i<1000:
        lrn_point=0.01
    elif i<2000:
        lrn_point = 0.001
    elif i < 5000:
        lrn_point = 0.0001
    else:
        lrn_point = 0.0001
```

Adaptive의 가장큰 단점

- 그레디언트의 축적으로 인해 param이 변하질 않는다.

- 그래디언트의 축적을 제한 하는 방법?

Adadelta

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$$\Delta\theta_t = -\frac{\boxed{RMS[\Delta\theta]_{t-1}}}{RMS[g]_t} g_t$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t.$$

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}.$$

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

1.learning rate term이 없어서 정할 필요가 없다.

RMS Prop

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

힐튼이 제시한 Optimizier
계산량이 적고 수식이 복잡하지 않아 많이 사용한다. 성능도 합리적이다.
기본적으로 추천하는 learning rate = 0.0001

ADAM
Adaptive Moment Esitimate

RMSProp에다가 momentum 값을 더하자

RMS

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

ADAM

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

Adam

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

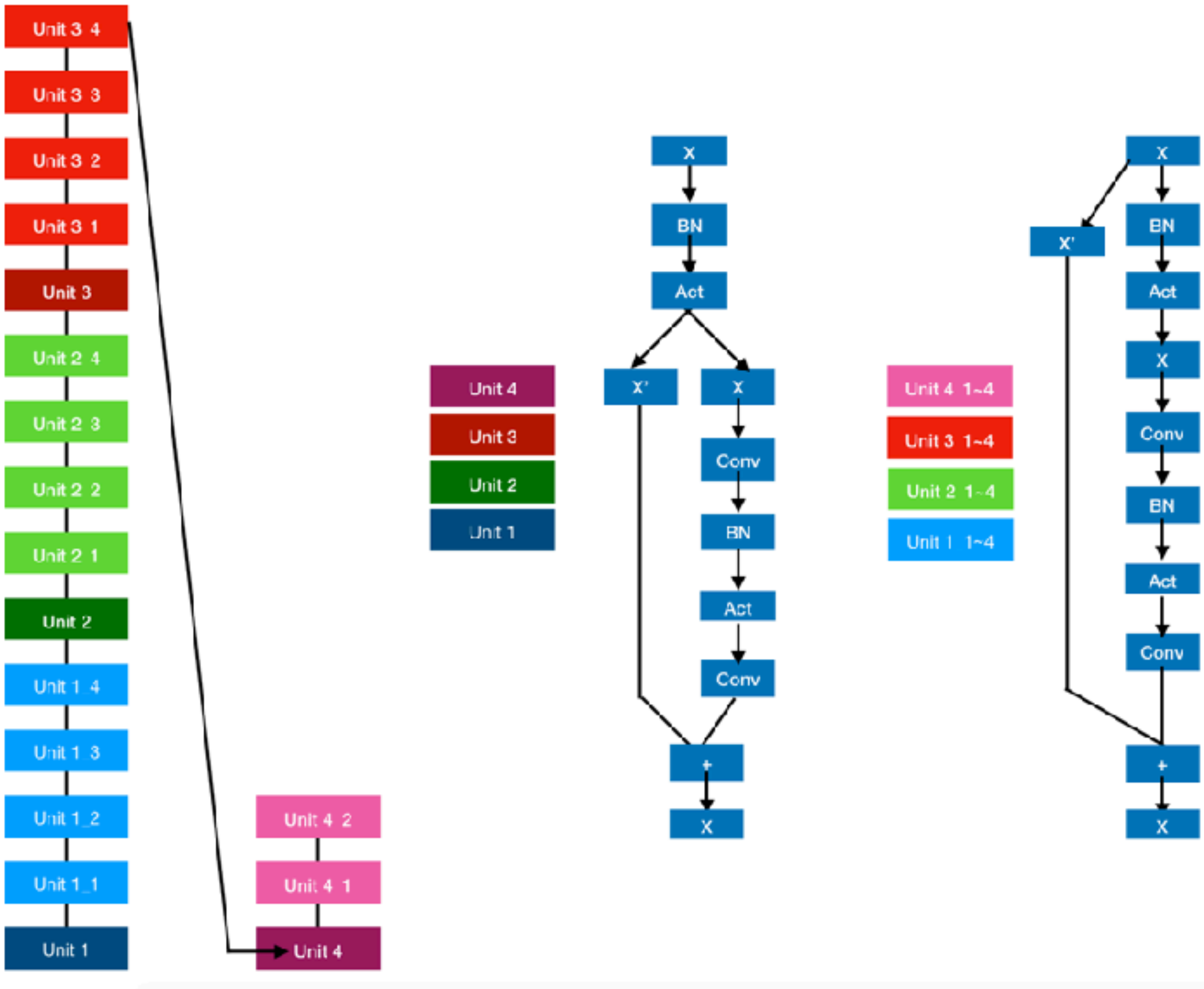
Like momentum , this sentence remember $_{\text{t-1's}}$ Gradient

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

To avoid over number of optimizer ,
Using mt^

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t.$$

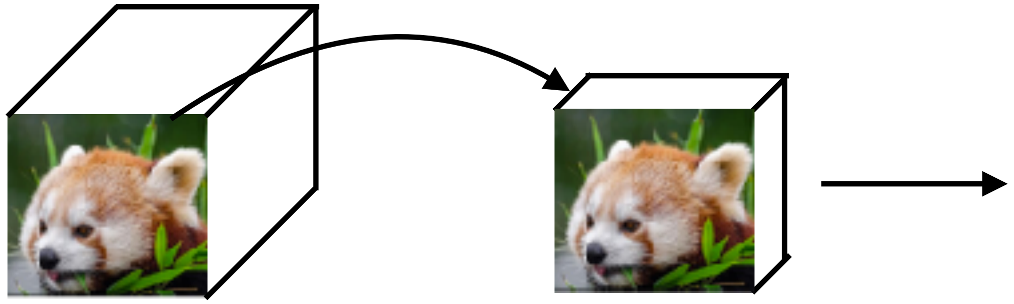
Wide resnet을 이용한 실습



Tensor flow

Batch Images Random Augmentation

```
image = tf.image.resize_image_with_crop_or_pad(image , image_size+4 , image_size+4)
image = tf.random_crop(image , [image_size , image_size ,3])
image = tf.image.random_flip_left_right(image)
image = tf.image.random_flip_up_down(image)
# Brightness / saturatio / constrast provides samll gains 2%~5% on cifar
image = tf.image.random_brightness(image , max_delta=63. / 255.)
image = tf.image.random_saturation(image , lower=0.5 , upper=1.8)
image = tf.image.per_image_standardization(image)
```



Random Augmentation

Wide resnet을 이용한 실습

- Random batch augmentation
- SGD , Momentum Optimizer
- Tensor board 을 이용한 Visualization