Deep Into Deep

김성찬

08 Training Neural Network II

Contents

1. Mini-batch SGD Optimization

For N Epochs{

For each training batch $\{(\mathrm{x}_b, y_b)\}_{b=1}^{B}$ {

$$\mathrm{w} \leftarrow \mathrm{w} - \alpha \frac{\partial}{\partial \mathrm{w}} L(\mathrm{w})$$

}

}

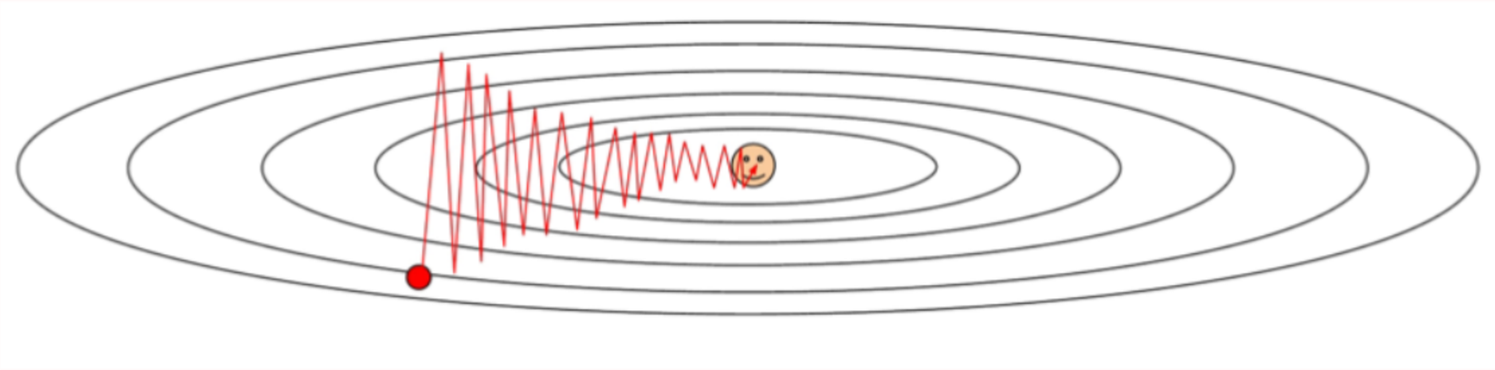
1. Sample a batch of data
2. Forward propagate it through the graph (network), get loss.
3. Backpropagate to calculate the gradients.
4. Update the parameters using the gradients.

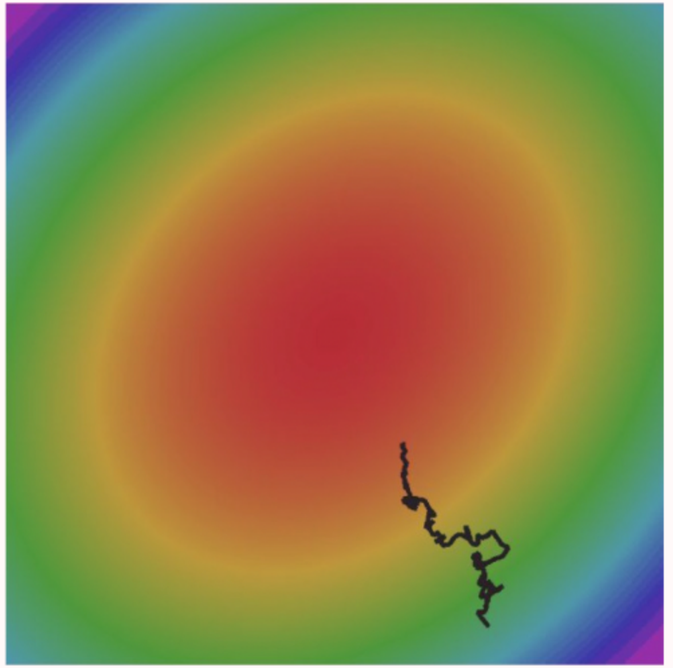
1. Mini-batch SGD Optimization

Problem with SGD

What if loss changes quickly in one direction and slowly in another?
What does gradient descent do?

-> Very slow progress along shallow dimension,
jitter along steep direction.





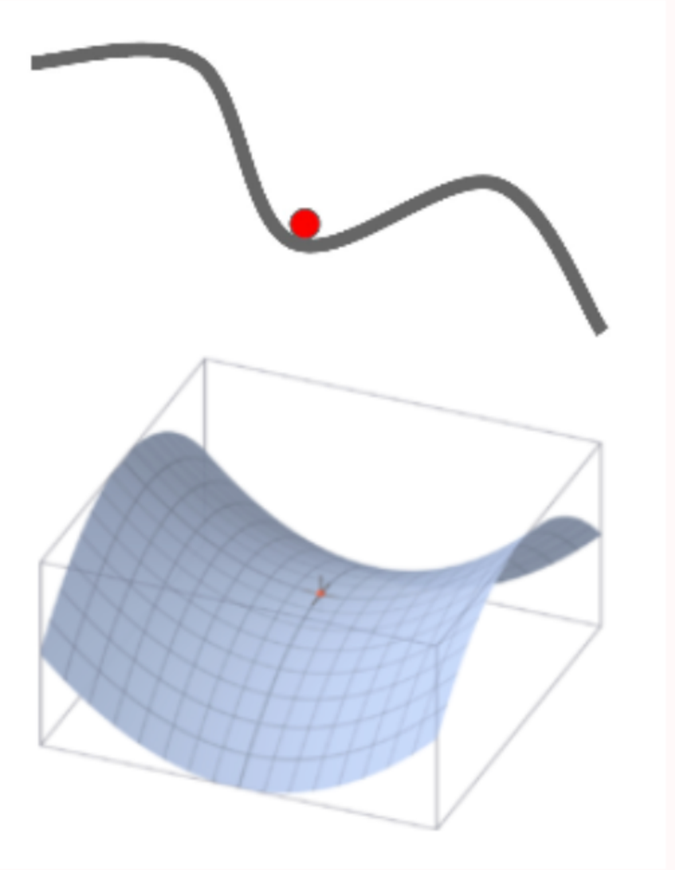
1. Mini-batch SGD Optimization

Problem with SGD

What if the loss function has a local minima or saddle point?

Zero gradient
-> Gradient descent gets stuck.

Saddle points much more common in high dimension.



2. SGD + Momentum

SGD

$$w_{t+1} = w_t - \alpha \nabla f(w_t)$$

SGD + Momentum

Introduce a concept of *velocity* v

$$-\nabla f(w_t) = \frac{v_{t+1} - v_t}{\alpha} \rightarrow v_{t+1} = v_t - \alpha \nabla f(x_t)$$

In a more general form,

$$\rightarrow v_{t+1} = \rho v_t - \alpha \nabla f(x_t)$$

손실 함수 f는 potential energy이고 f의 기울기는 가속도다.

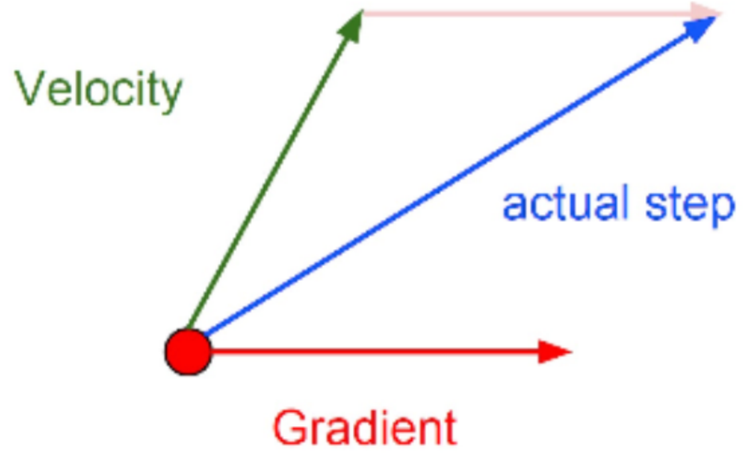
이때 가중치는 위치가 된다.

2. SGD + Momentum

SGD + Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t)$$

$$w_{t+1} = w_t + v_{t+1}$$

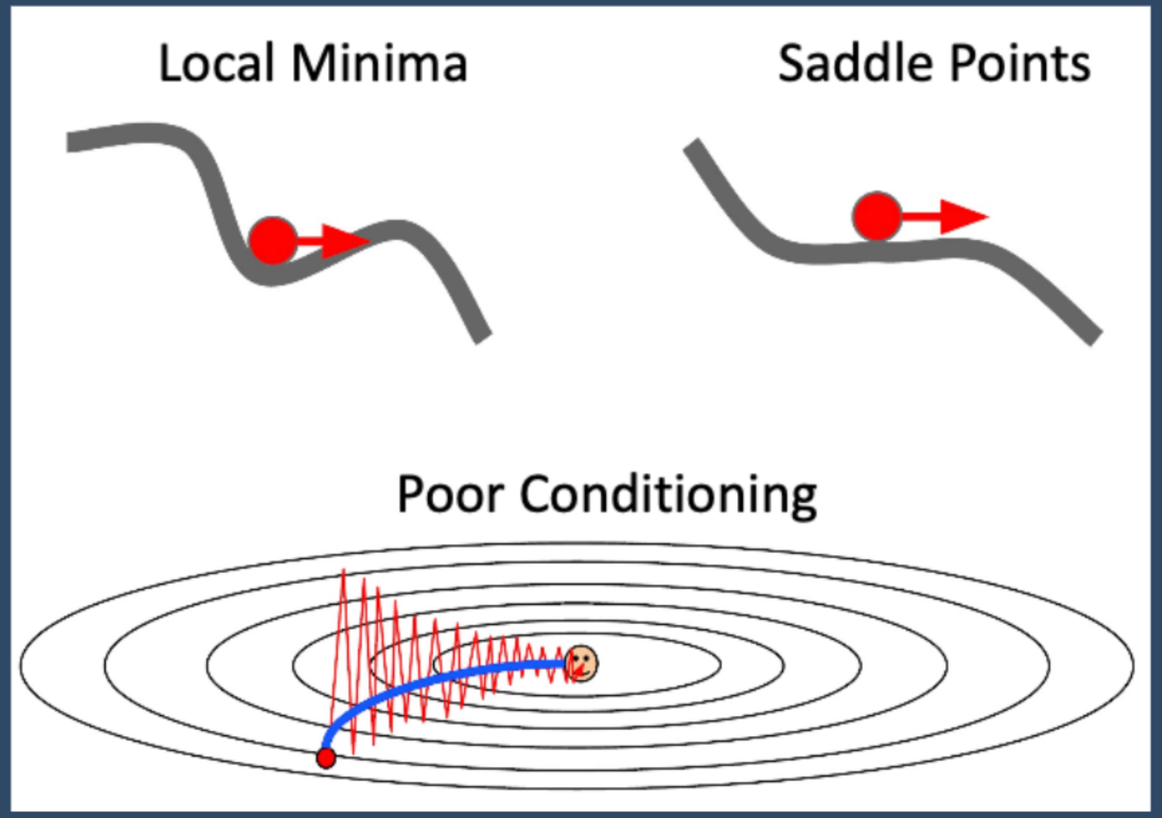


속도의 개념을 사용한다.

+ 마찰력에 해당하는 값을 준다.

2. SGD + Momentum

SGD + Momentum이 SGD의 문제를 해결할 수 있다.

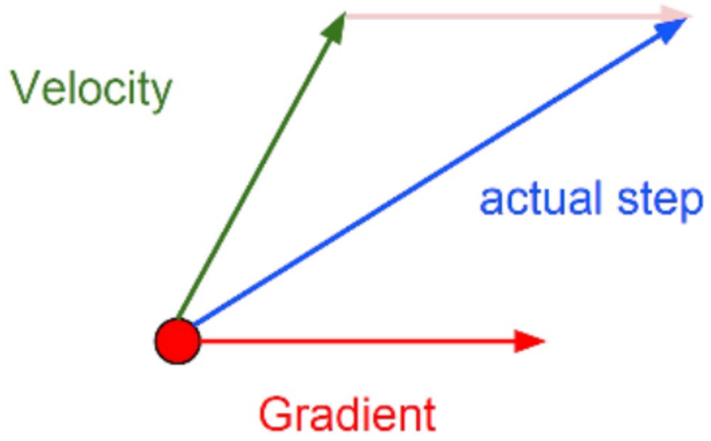


3. Nesterov Momentum

SGD + Momentum

$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t)$$

$$w_{t+1} = w_t + v_{t+1}$$

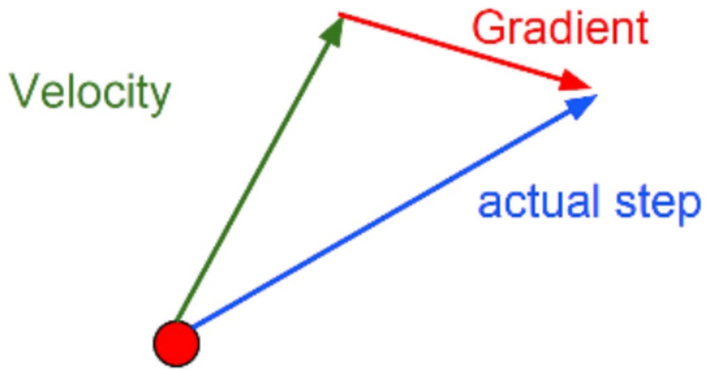


Combine gradient at current point with velocity to get step used to update weights

Nesterov Momentum

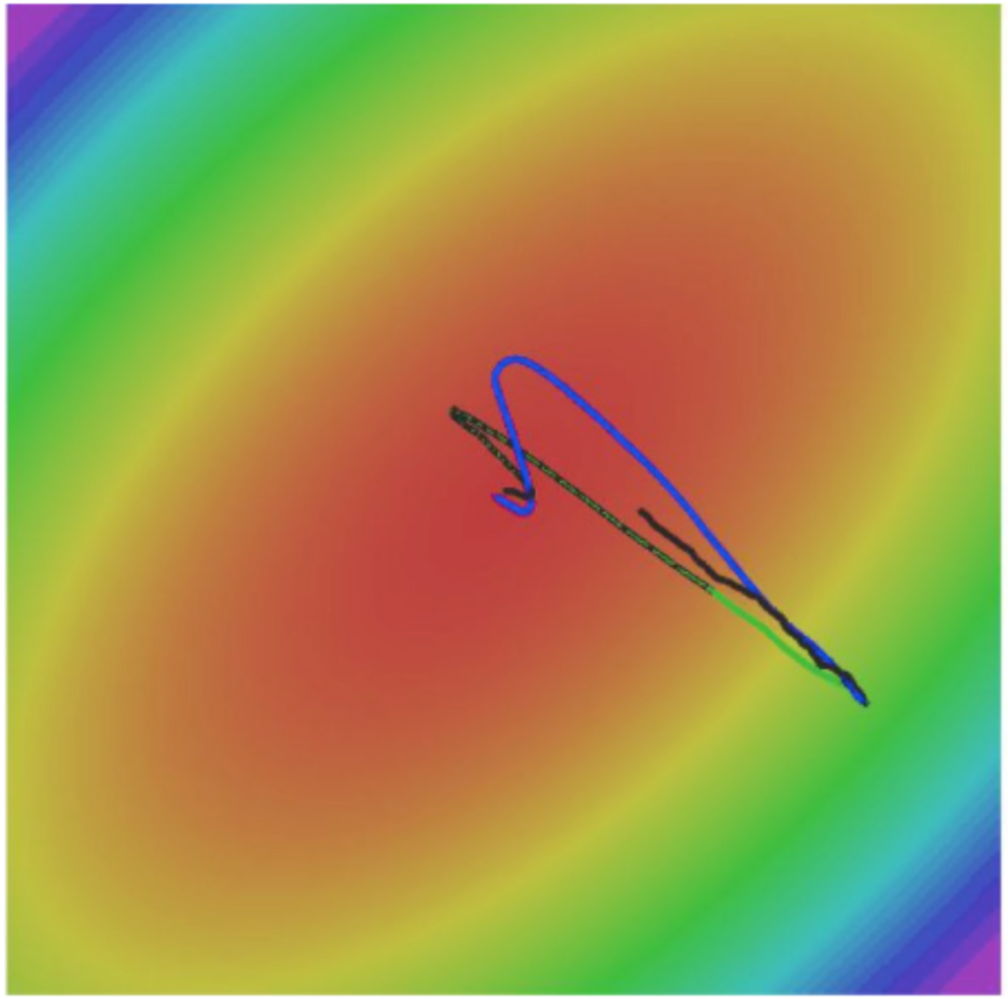
$$v_{t+1} = \rho v_t - \alpha \nabla f(w_t + \rho v_t)$$

$$w_{t+1} = w_t + v_{t+1}$$



"Look ahead" to the point where updating using velocity would take us; compute gradient there and mix it with velocity to get actual update direction

3. Nesterov Momentum

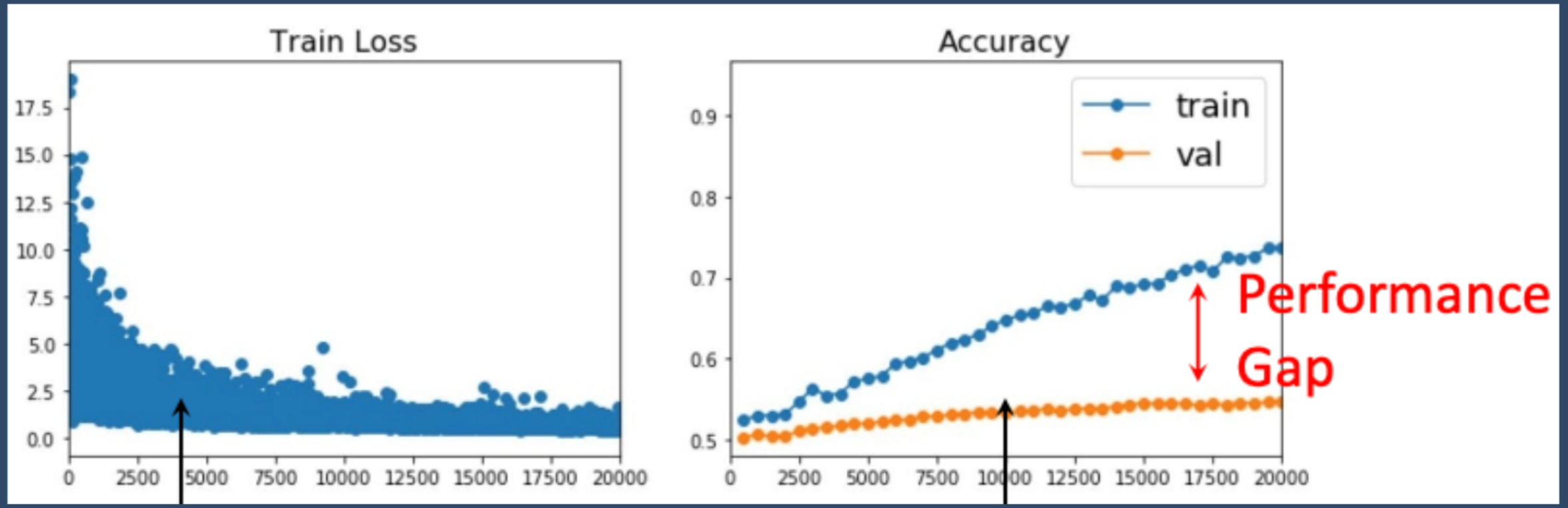


SGD

SGD + Momentum

Nesterov Momentum

4. Improve Test Error

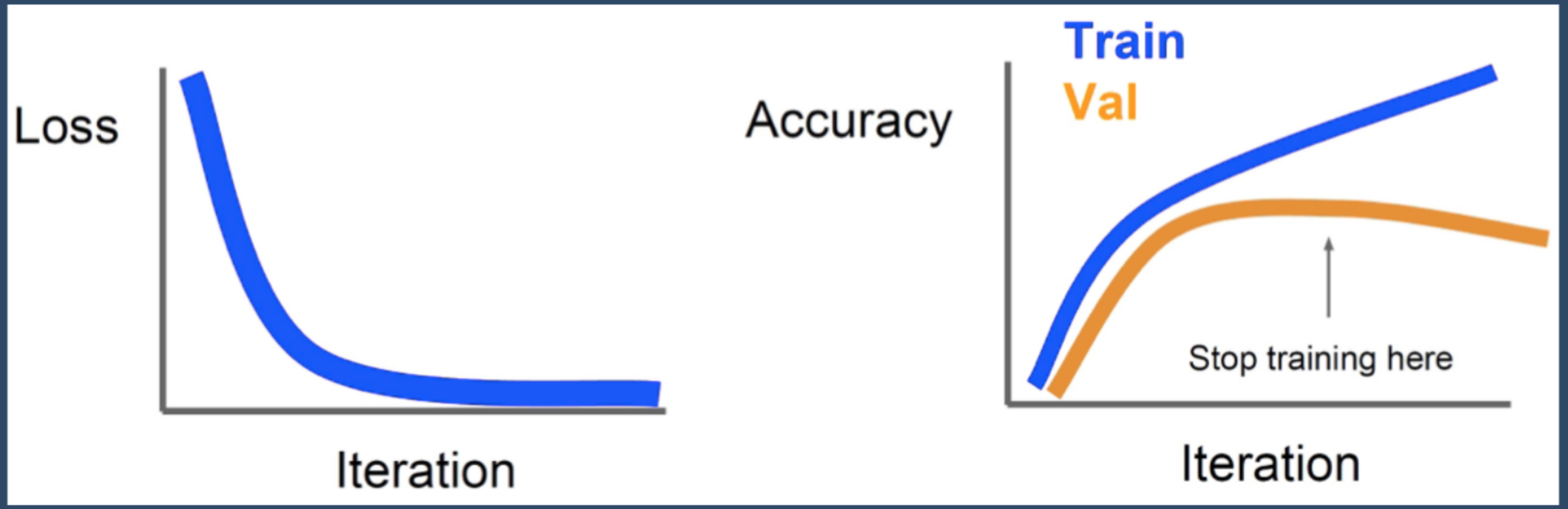


좋은 최적화 알고리즘을 통해 학습 에러를 줄일 수 있다.
하지만 새로운 데이터에 대한 에러는 어떻게 더 줄일 수 있을까?

4. Improve Test Error: Early Stopping

validation set에 대한 정확도가 감소하기 시작할 때 모델 학습을 멈추거나,

모델을 오랫동안 학습하면서 더 나은 모델이 있으면 model snapshot을 저장한다.



4. Improve Test Error: Model Ensembles

여러 개의 독립적인 모델을 학습시킨다.

Test-time 시 여러 개의 모델이 내놓는 결과의 평균을 사용한다.

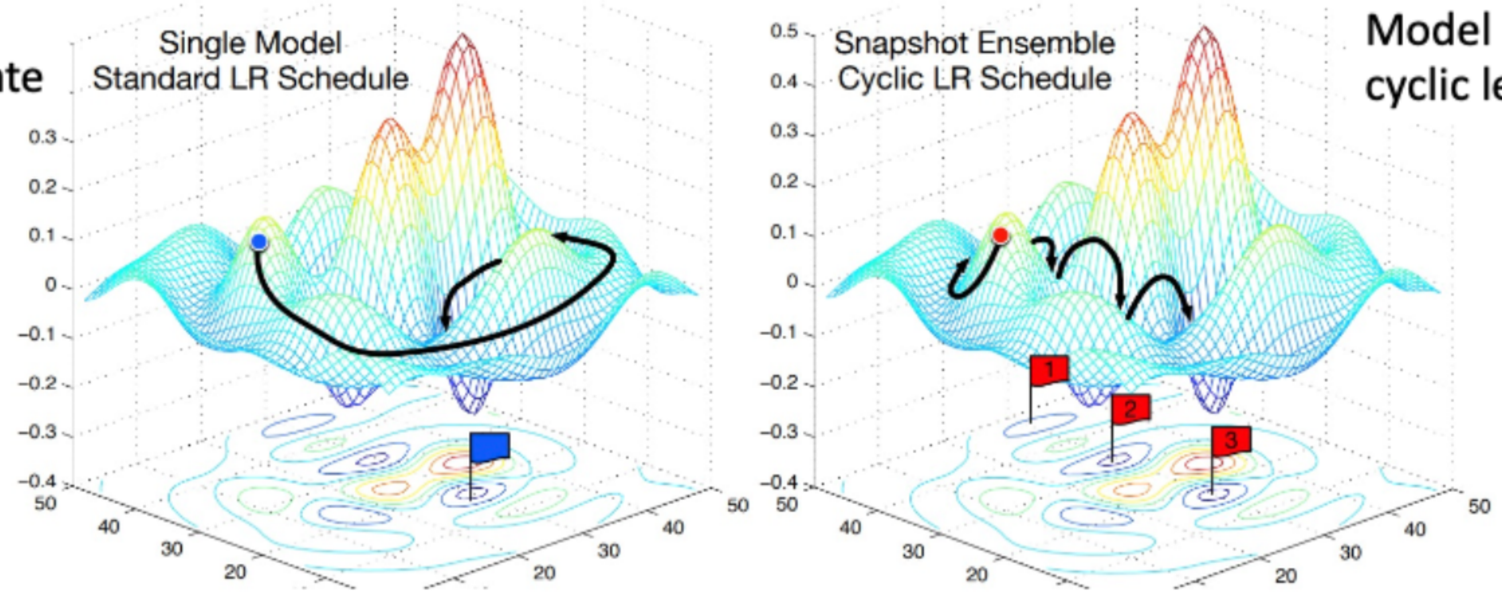
예를 들어, 10개의 독립적인 모델을 학습시킨 후에 강아지 사진에 대해 예측했을 때, 7개의 모델은 "강아지"라고 예측하고 3개의 모델은 "고양이"라고 예측하면, "강아지"로 예측한 것으로 하는 것이다.

이를 통해 2% 정도의 성능을 올릴 수 있다.

4. Improve Test Error: Model Ensembles

- Instead of training independent models,
  use multiple snapshots of a single model during training!

Single model with standard learning rate



Model ensembles with cyclic learning rate

단일 모델의 성능을 향상시키는 방법은?

Regularization

5. Regularization - A Common Pattern

학습 시: randomness를 추가하여 학습한다.

테스트 시: randomness의 평균을 구하여 사용한다.

$$y = f(x, z)$$

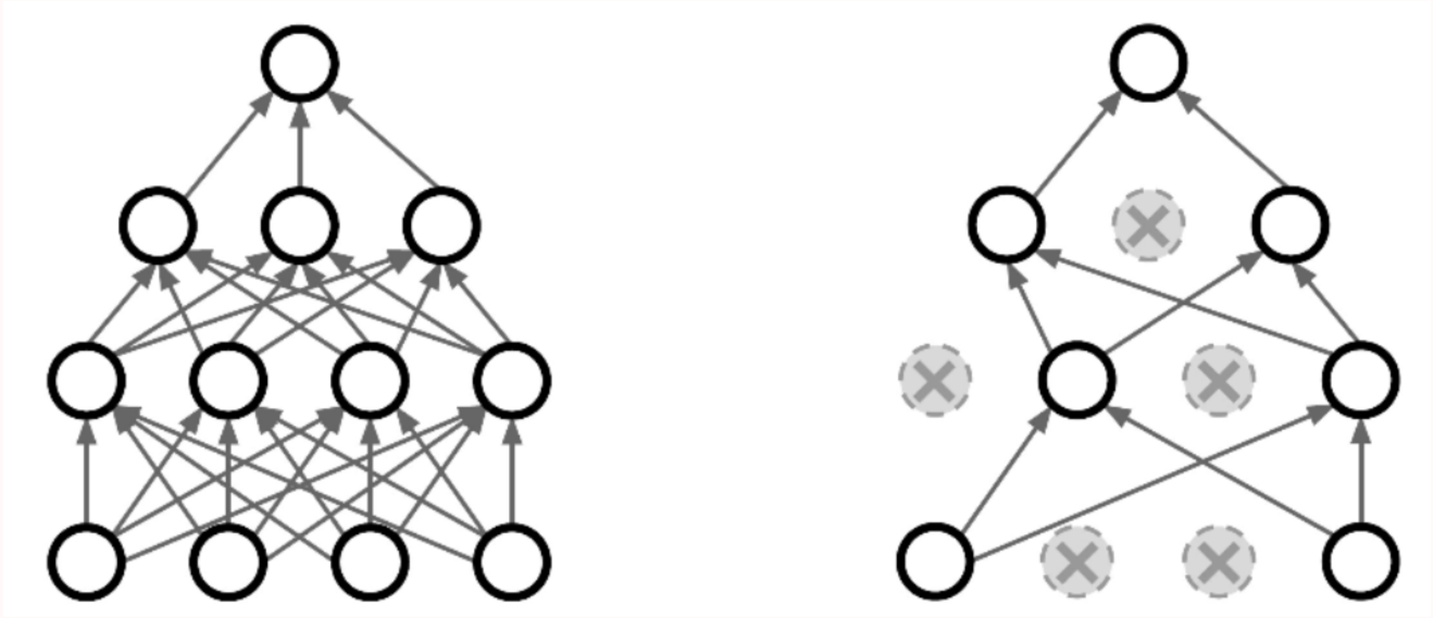
Output (e.g., label) | Input (e.g., Image) | Random variable

$$y = f(x) = E_z[f(x, z)]$$
$$= \int f(x, z)p(z)dz$$

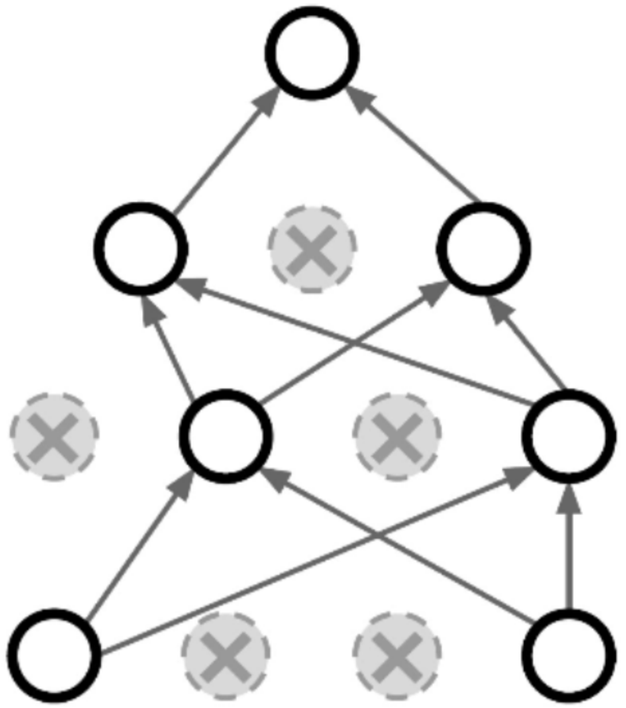
5. Regularization: Dropout

Dropout은 학습하는 동안 overfitting이 발생하는 것을 피하기 위해 일반적으로 사용되는 방법이다.

- In each forward pass, randomly set some neurons to zero.
- Probability of dropping is a hyper-parameter; 0.5 is common.



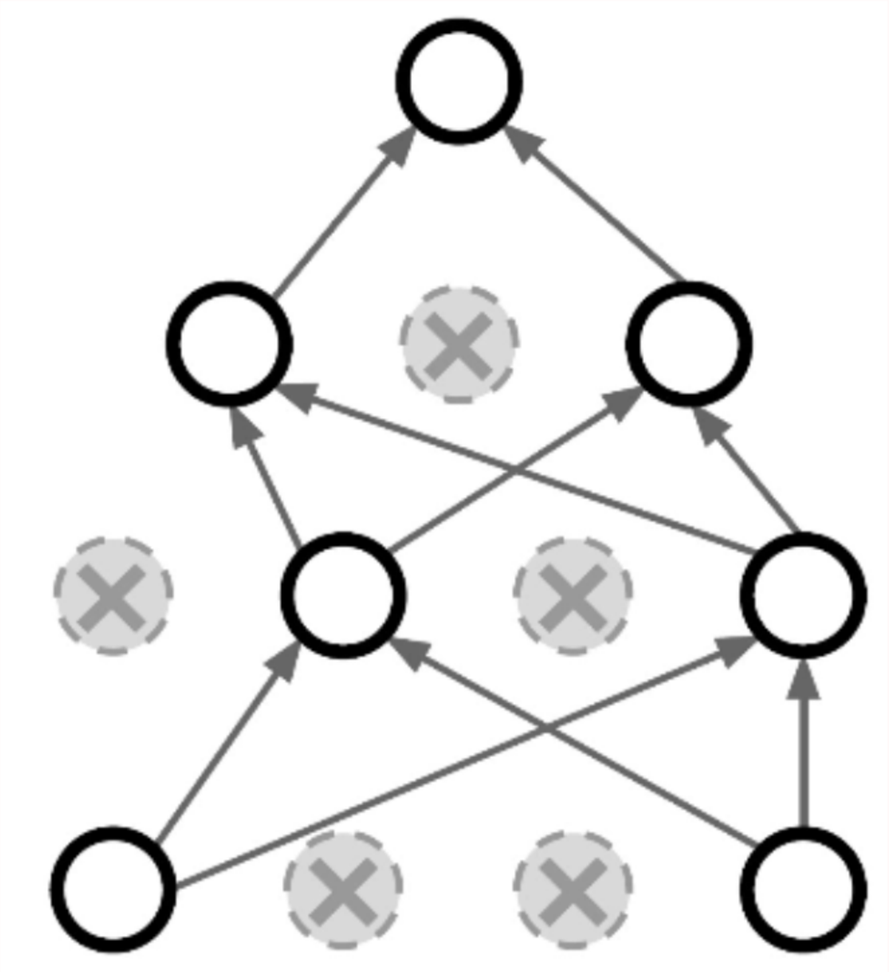
5. Regularization: Dropout



Forces the network to have a redundant representation;
Prevents co-adaptation of features

representation learning이 더 잘 이루어질 수 있다!!

5. Regularization: Dropout



다른 해석으로는

Dropout은 파라미터를 공유하는 큰 ensemble 모델을 학습하는 것과 같다.

5. Regularization: Dropout

학습하는 동안, Dropout을 통해 randomness를 추가할 수 있다.

$$\mathrm{y} = f(\mathrm{x}, \mathrm{z})$$

Output (e.g., label) | Input (e.g., Image) | Random mask

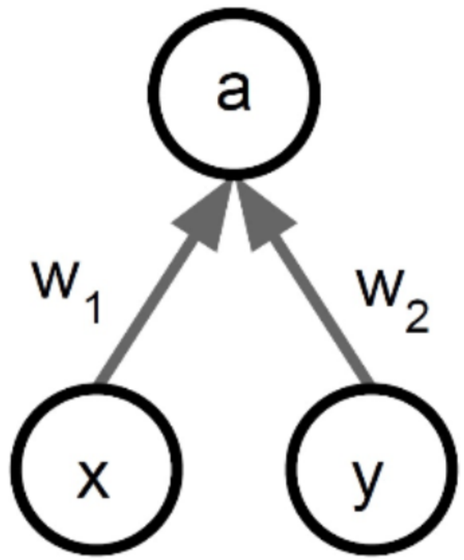
이 randomness의 평균을 구하는 것은 model ensemble과 유사하다.

$$\mathrm{y} = f(\mathrm{x}) = E_{\mathrm{z}}[f(\mathrm{x}, \mathrm{z})] = \int f(\mathrm{x}, \mathrm{z}) p(\mathrm{z}) d\mathrm{z}$$

하지만, 실제로 randomness의 평균을 구하는 것은 어렵다.

5. Regularization: Dropout

위 integral을 근사하려면



Consider a single neuron.

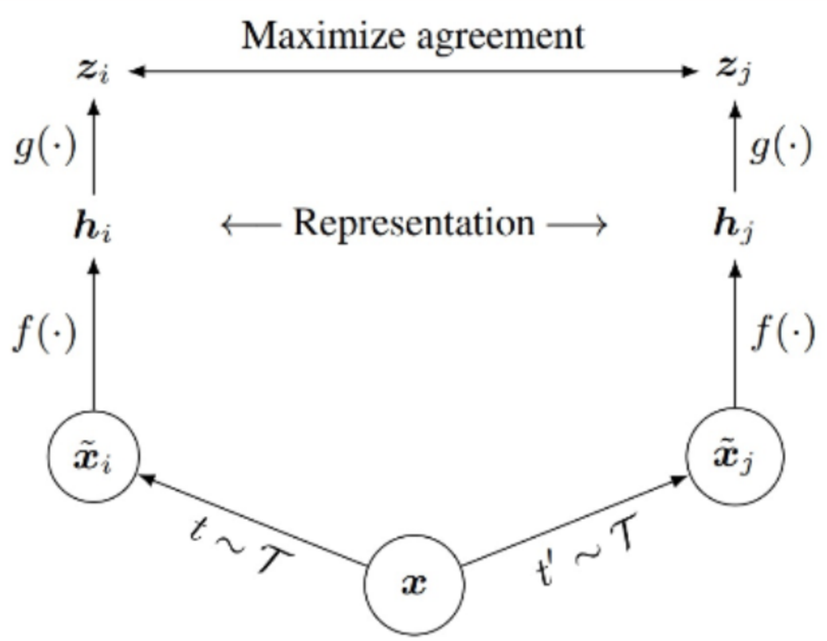
At test time an original output is: $y = w_1 x_1 + w_2 x_2$

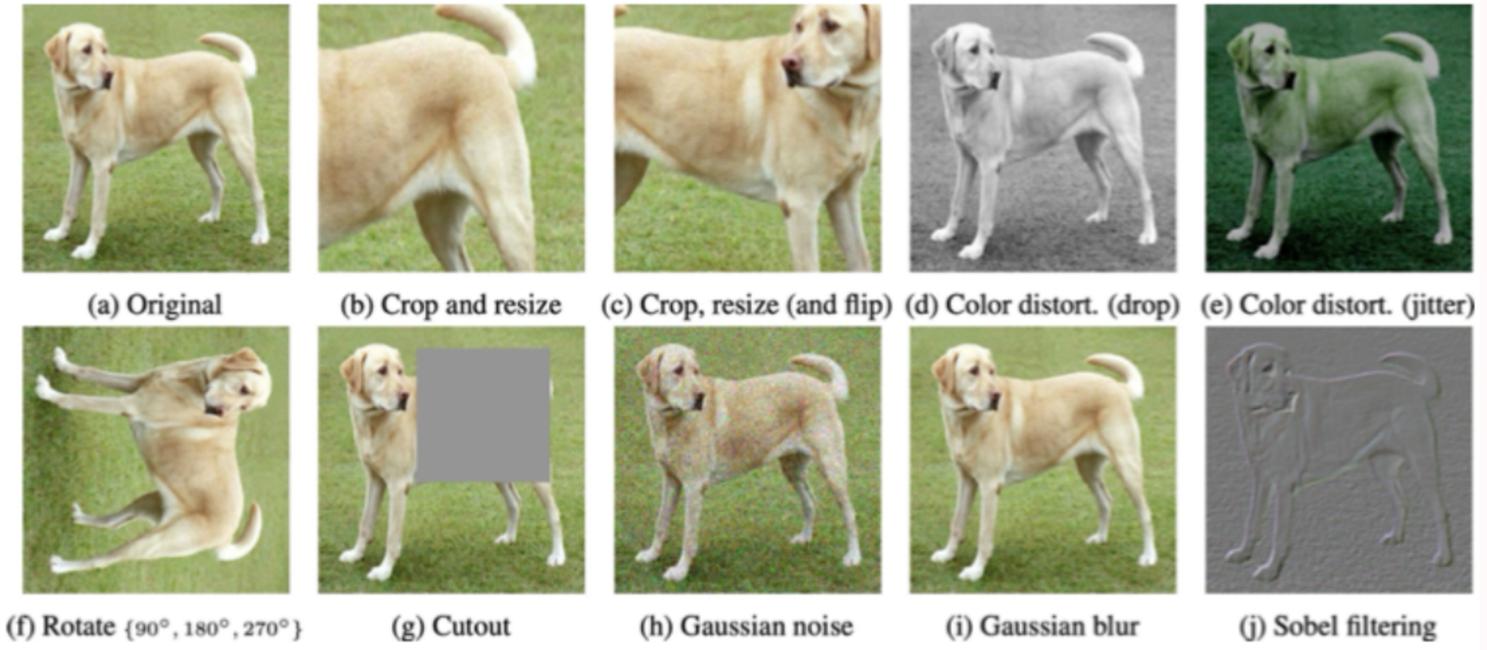
When the dropout is used with $p = 0.5$,
we have

$$y = f(x) = E_z[f(x,z)] = \frac{1}{4}(w_1 x_1 + w_2 x_2) + \frac{1}{4}(w_1 x_1 + w_2 0) + \frac{1}{4}(w_1 0 + w_2 0) + \frac{1}{4}(w_1 0 + w_2 x_2) = \frac{1}{2}(w_1 x_1 + w_2 x_2)$$

At test time, **multiply the original output** by dropout probability to compute the average

5. Regularization: Data Augmentation for Representation Learning



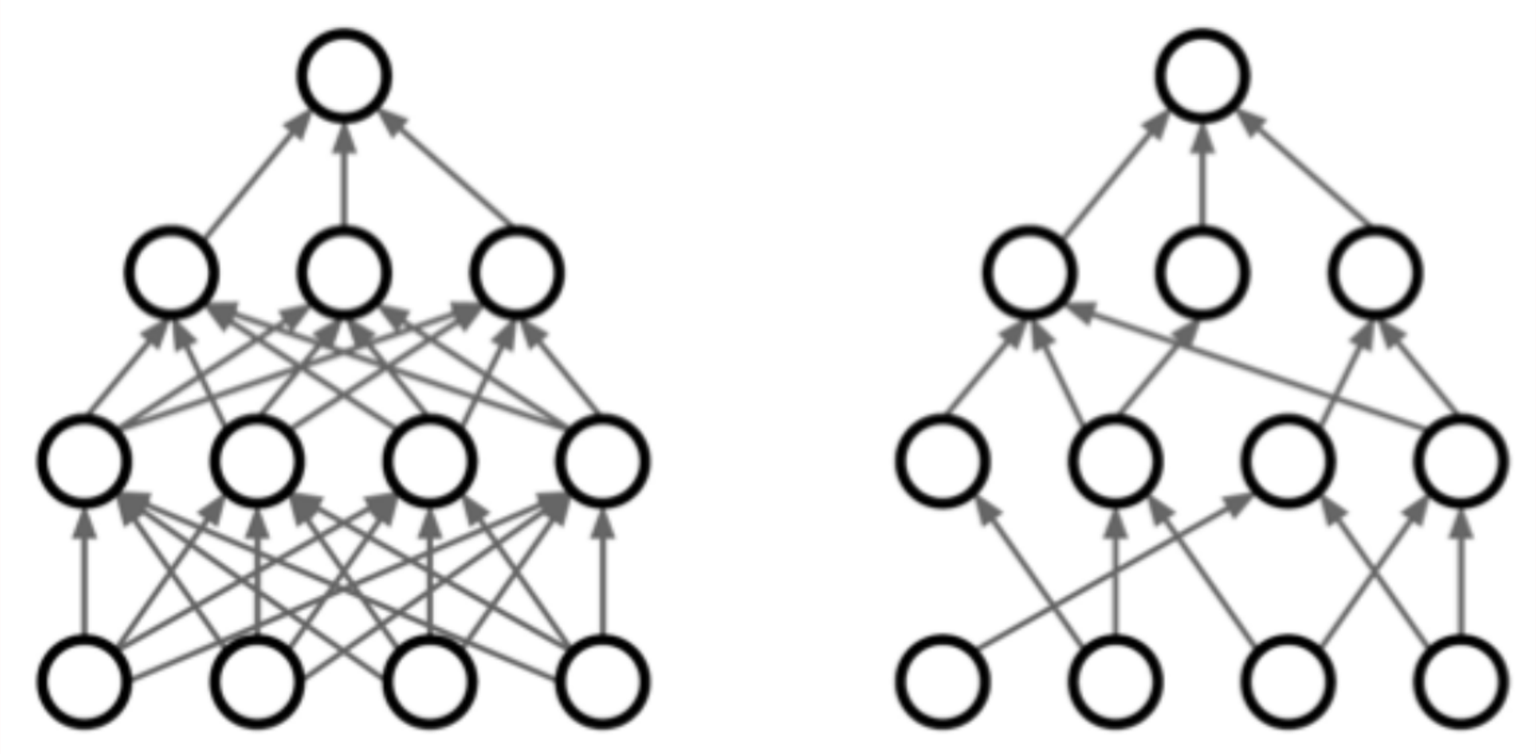


같은 이미지로부터 서로 다른 변형을 통해 만들어진 두 이미지는 같은 "의미"를 가져야 한다.

의미 == Representation

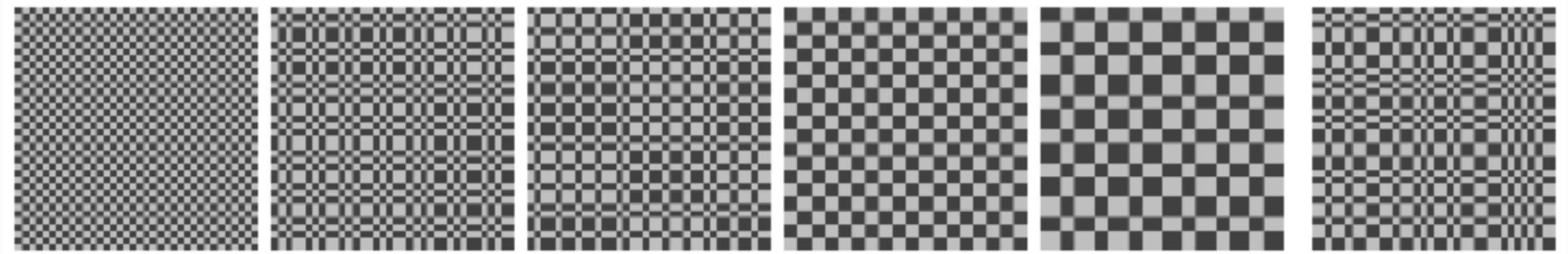
5. Regularization: DropConnect

Dropout은 노드를 drop하지만
DropConnect는 노드끼리의 연결, 가중치를 drop한다.



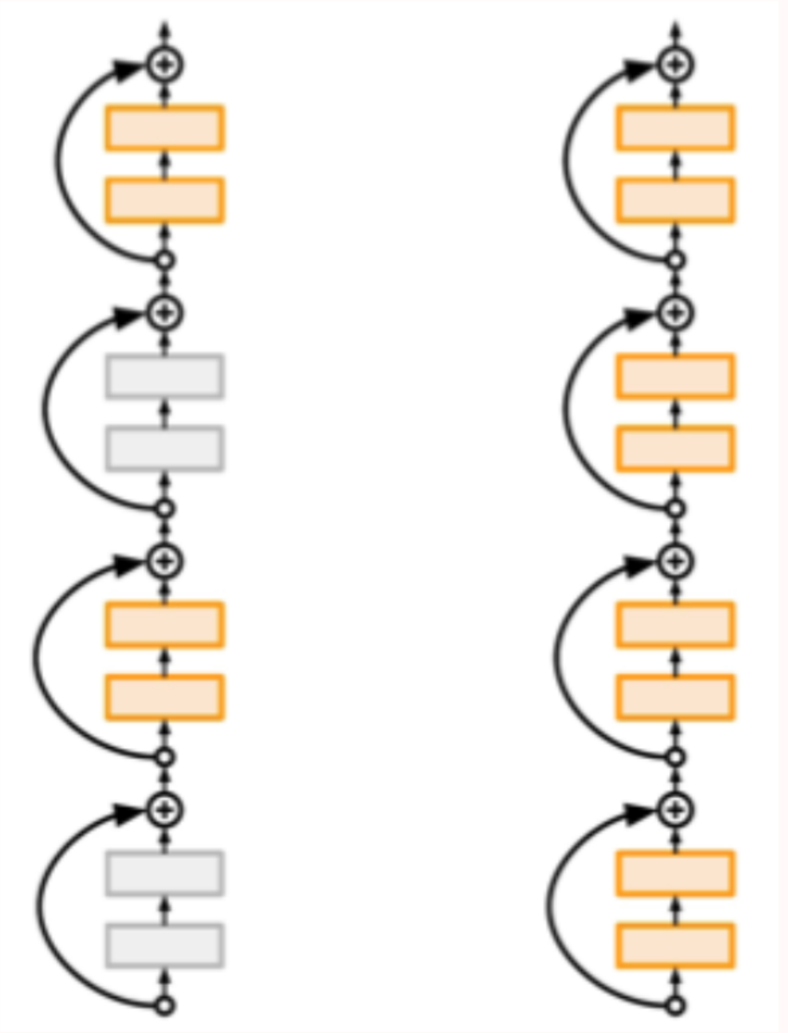
5. Regularization: Fractional Pooling

Fractional Max Pooling: Max Pooling을 적용하는 kernel_size가 모두 제각각이다.

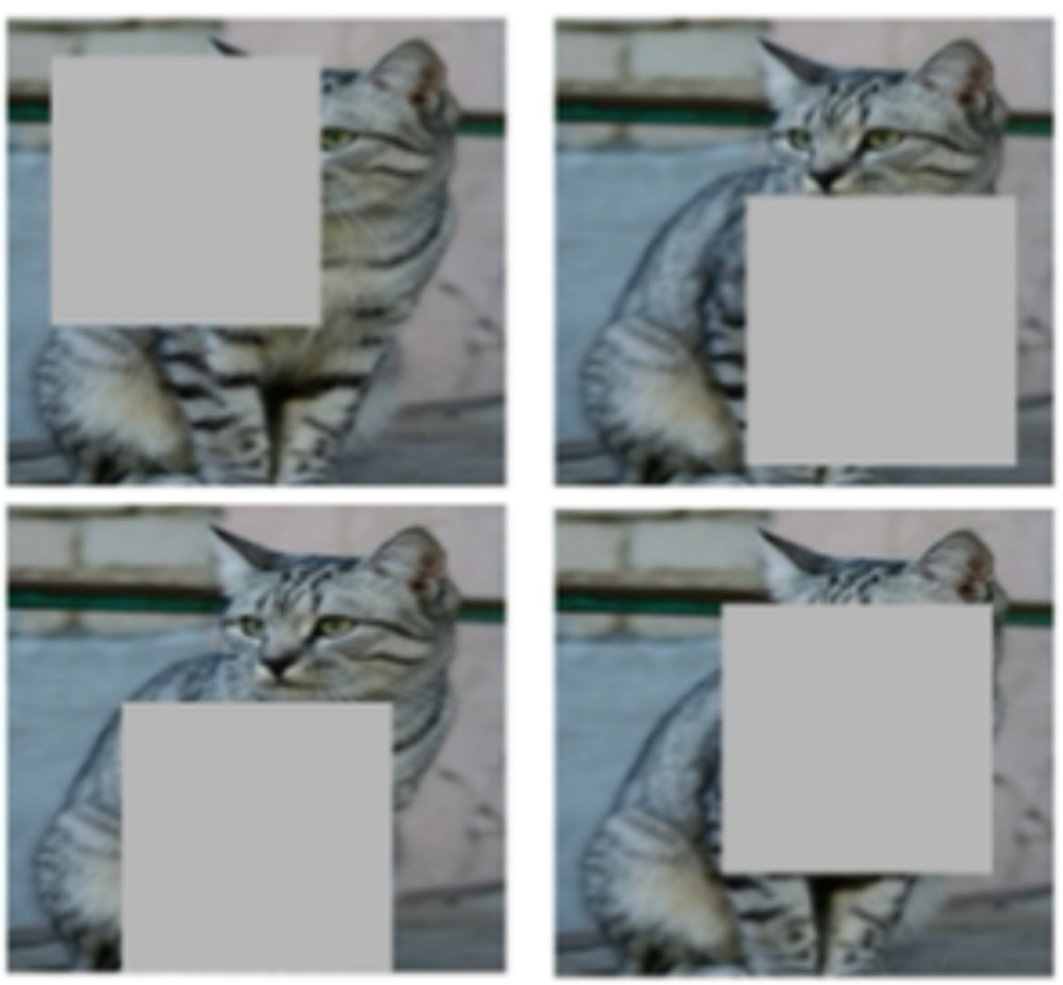


5. Regularization: Stochastic Depth

Training: 무작위로 layer를 skip한다.
Testing: 모든 네트워크를 사용한다.



5. Regularization: Cutout



Training:
랜덤으로 이미지의 특정 부위를 0으로 만든다.

Testing:
원본 이미지를 사용한다.

5. Regularization: Cutout

Training: 무작위로 섞은 이미지에 대해 학습한다.
Testing: 원본 이미지를 사용한다.

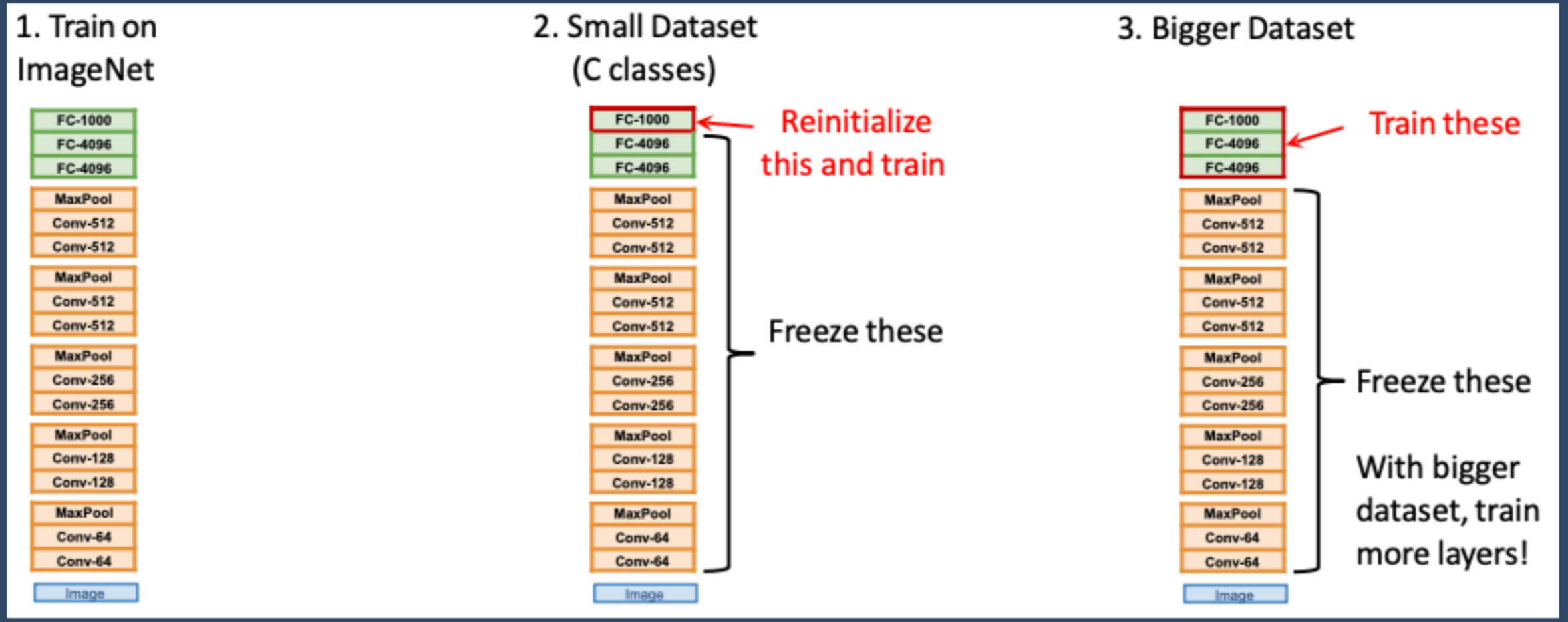
CNN

Target label:
cat: 0.4
dog: 0.6

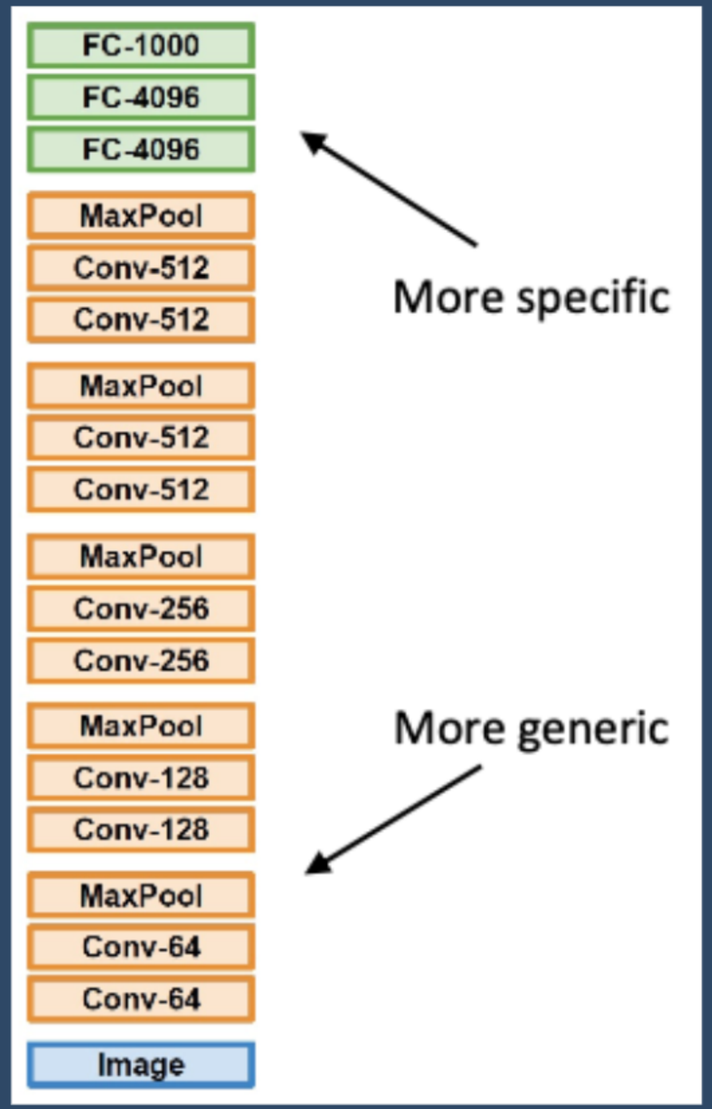
Randomly blend the pixels
of pairs of training images,
e.g. 40% cat, 60% dog

6. Transfer Learning with CNNs

Pre-train -> Fine-tuning

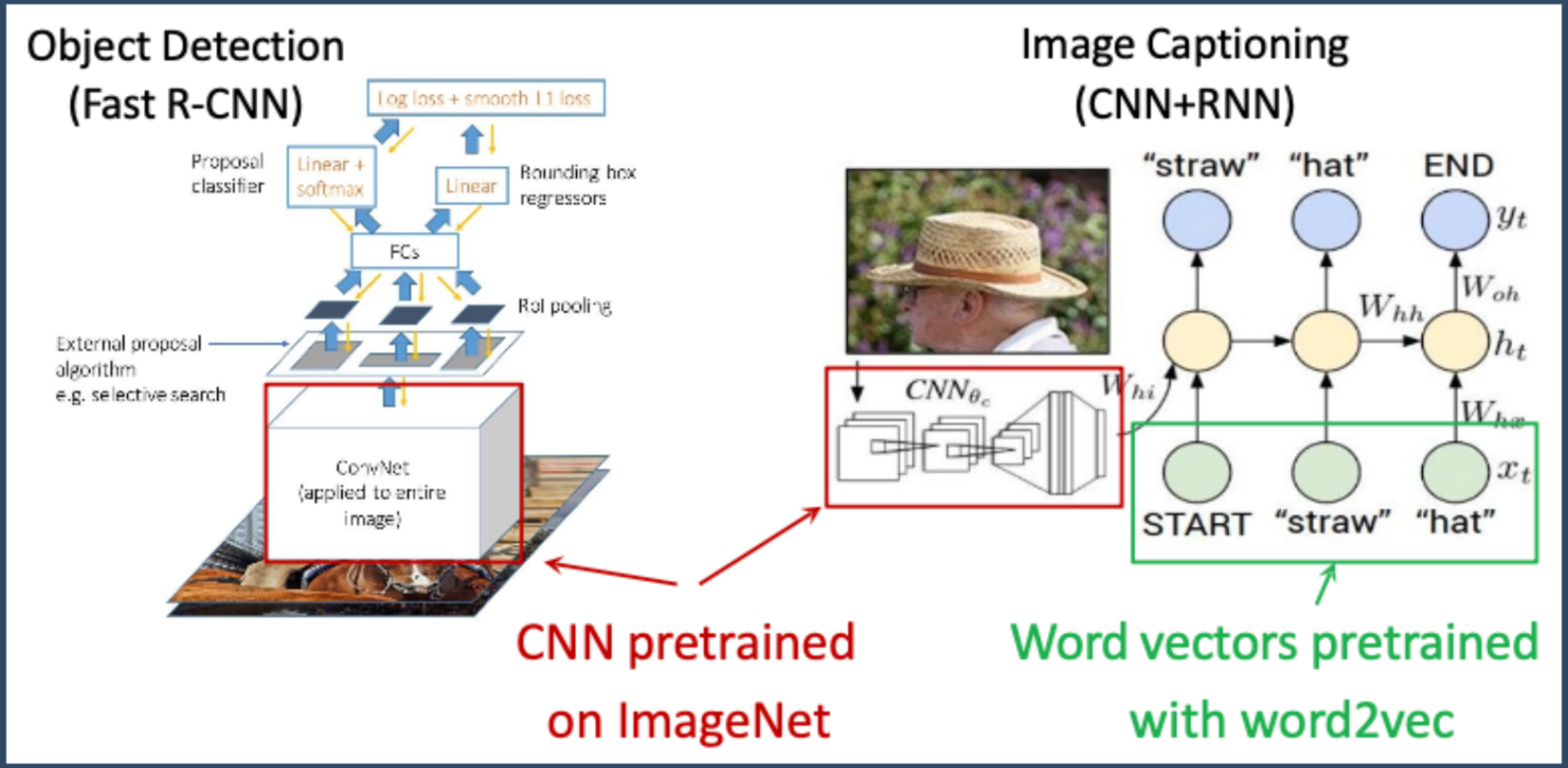


6. Transfer Learning with CNNs



	very similar dataset	**very different dataset**
very little data	Fine-tune linear classifier on top layer	You're in trouble... Try linear classifier from different stages
lots of data	Fine-tune a few layers	Fine-tune a larger number of layers

6. Transfer Learning with CNNs



Thank you for listening

Deep Into Deep