

Improving Generalization in Neural Networks

Marleen de Bruijne
5 March 2021
Elements of Machine Learning

UNIVERSITY OF COPENHAGEN



1

UNIVERSITY OF COPENHAGEN

Improving generalization

The cartoon consists of two panels. In the first panel, a male stick figure points to a whiteboard with the equation $\int x^2 = \pi$ and says, "WOW, YOU SUCK AT MATH." In the second panel, a female stick figure points to the same whiteboard with the same equation and says, "WOW, GIRLS SUCK AT MATH." The cartoon illustrates the concept of generalization by showing how a single example (a male student) is used to draw a conclusion about a larger group (all students), which is then applied to a different group (girls).

Or: How not to draw the wrong conclusions from small samples

2

General remarks

- We are recording
- Please ask questions!
 - I cannot see you when my lecture is up, so:
 - Unmute and say something
 - Or type in the chat (but I may not always see this)

3

This lecture

- Recap: Generalization, underfitting, and overfitting
- Strategies to improving generalization

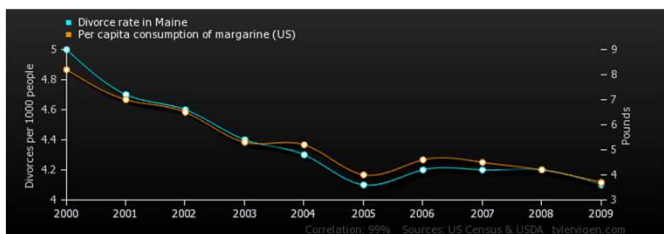
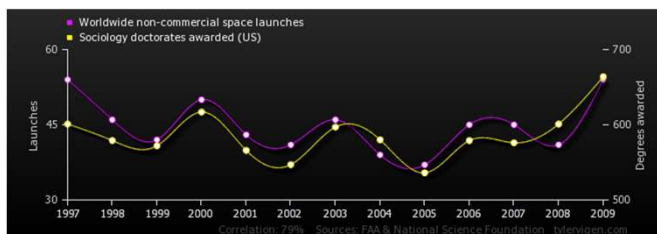
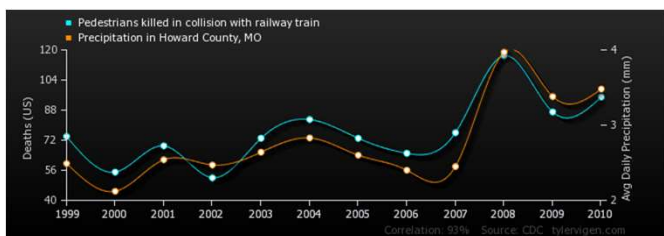
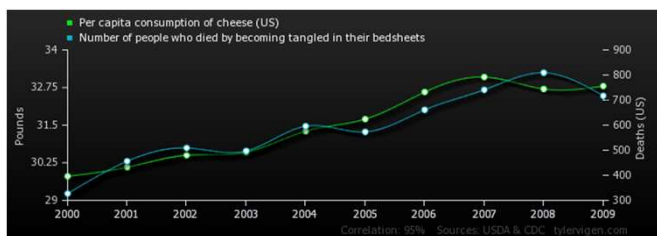
4

Generalization

- Generalization = performance on unseen data
- A classifier/regressor that performs well on the training data may not perform well on new data, even if that data comes from the same data generating distribution
- Why not?

5

Not all correlations are meaningful ... especially not in small samples



Spurious Correlations --- <https://tylervigen.com/>

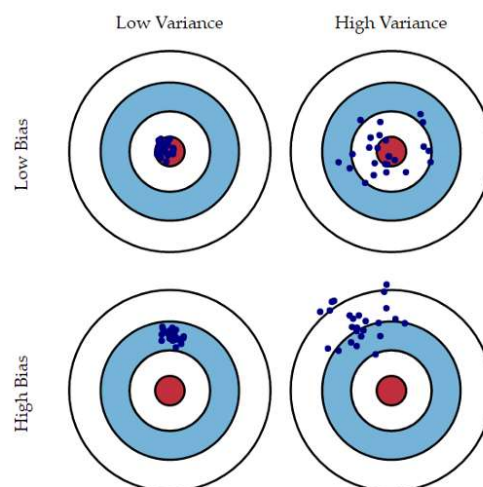
6

Are small samples a problem in practice?

- **In most cases, yes!**
- From the Deep learning book, Chapter 1:
 - "As of 2016, a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category, and **will match or exceed human performance when trained with a dataset containing at least 10 million labeled examples**. Working successfully with datasets smaller than this is an important research area, focusing in particular on how we can take advantage of large quantities of unlabeled examples, with unsupervised or semi-supervised learning."
- Not sure this rule of thumb is reasonable – we've seen excellent results with much fewer cases
- But need to be careful of risk of overfitting

7

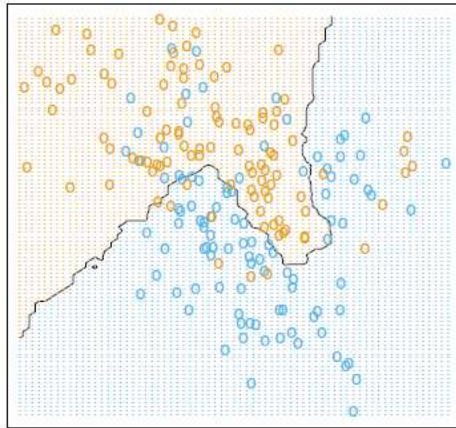
Given a specific model, smaller sample sizes will give have larger variance in the predictions



8

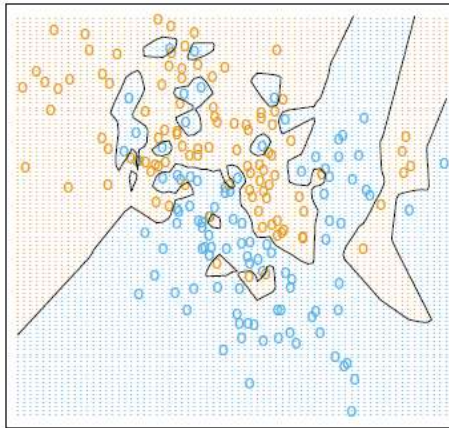
Example from lecture 1 – which one is overfitting?

$K=15$



*Regularized
Lower complexity
High bias?
Low variance*

$K=1$

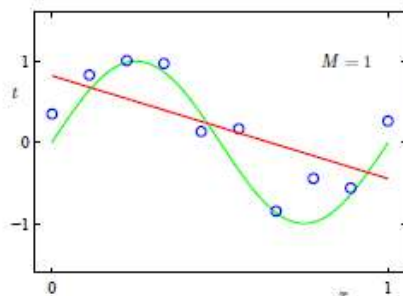


*Possibly
overfitting*

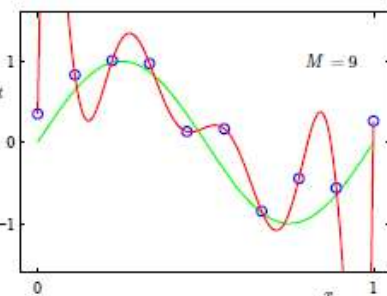
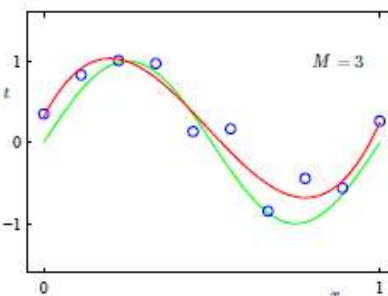
*Not regularized
High complexity
Low bias
High variance*

9

Or in the regression case – fitting a polynomial



*Underfitting
Low complexity
High bias
Low variance*

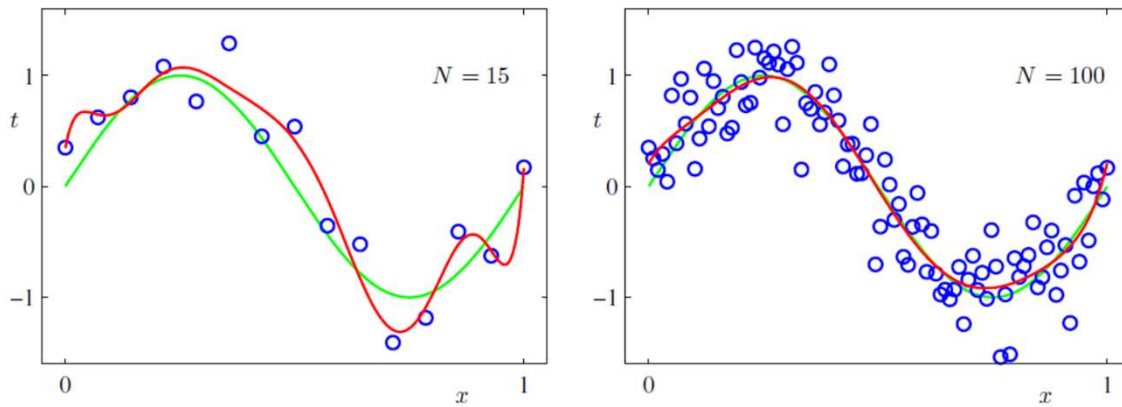


*Overfitting
High complexity
Low bias
High variance*

- Underfitted models make errors on both training data and unseen data
- Overfitted models make errors on unseen data

10

The overfitting problem reduces with more data



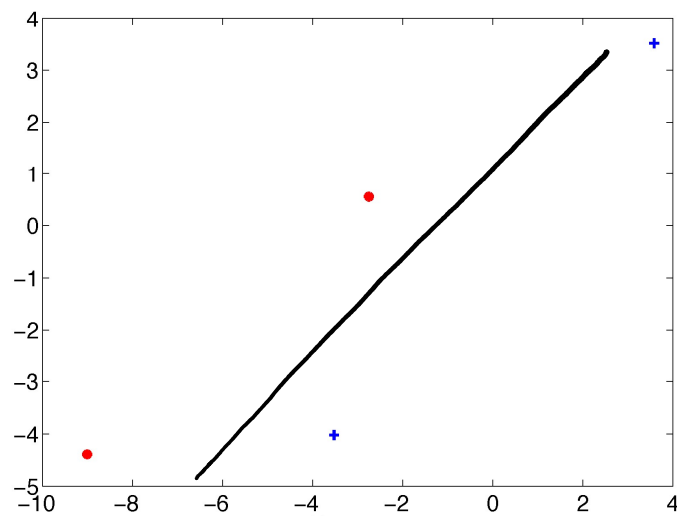
11

Generalization error depends on

- The data (distribution, class overlap, ...)
- Size of training data
- Model complexity

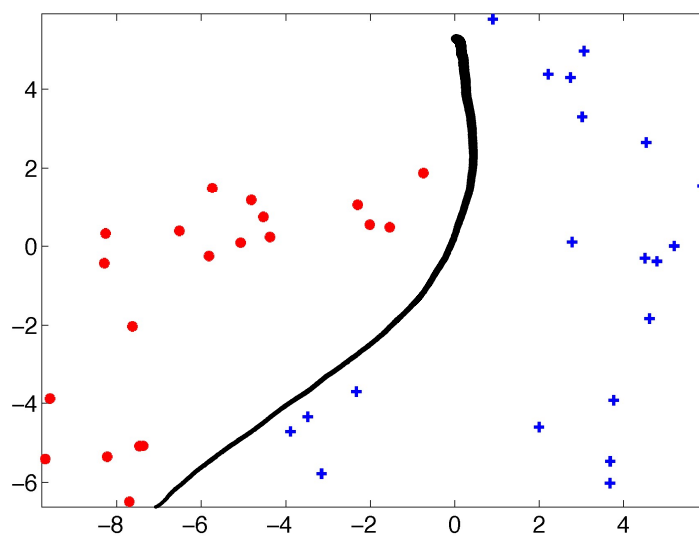
12

What classifier should we use?



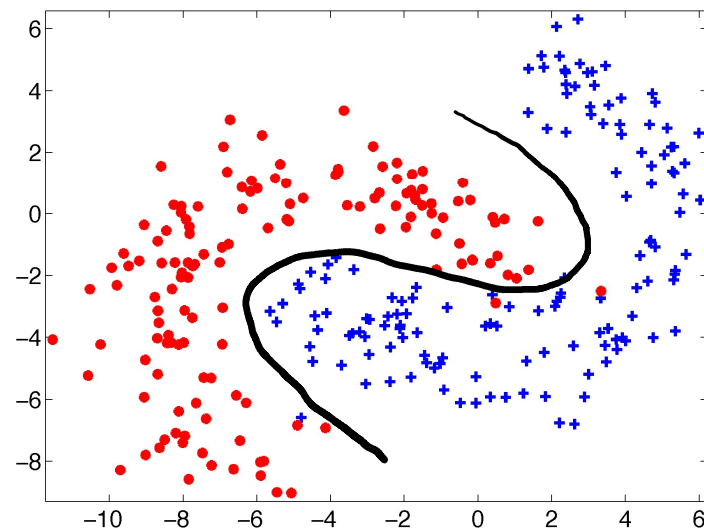
13

What classifier should we use?



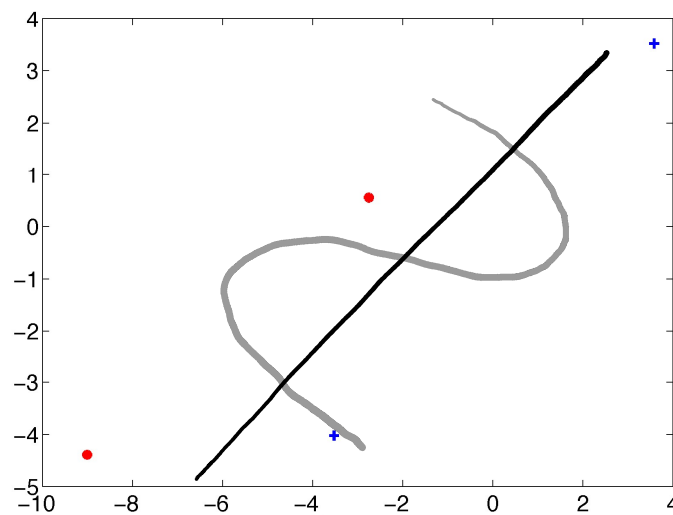
14

What classifier should we use?



15

What classifier should we use?



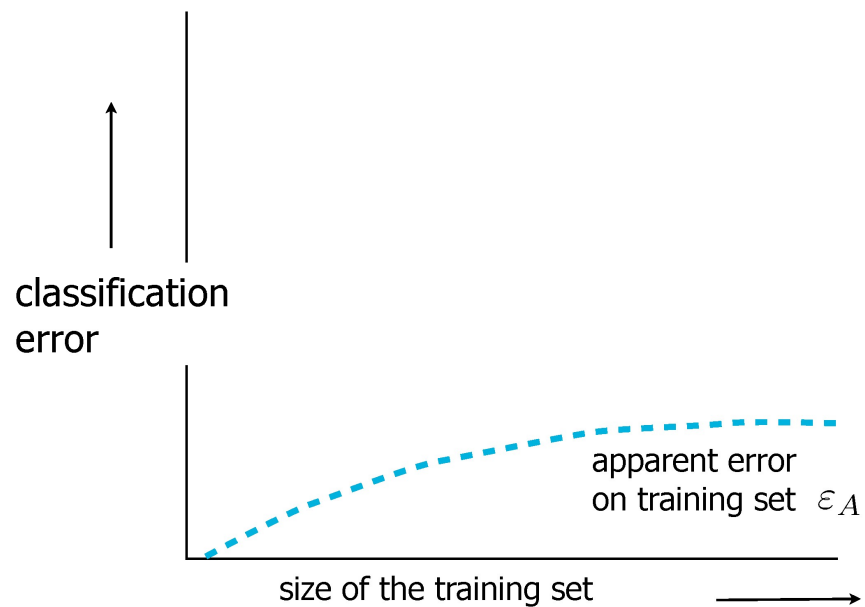
16

The problem of model selection

- What is the correct model depends on the underlying distribution
- In practice, we never know the underlying distribution, unless we have a problem that does not need solving
- Choose the simplest model that describes the data well

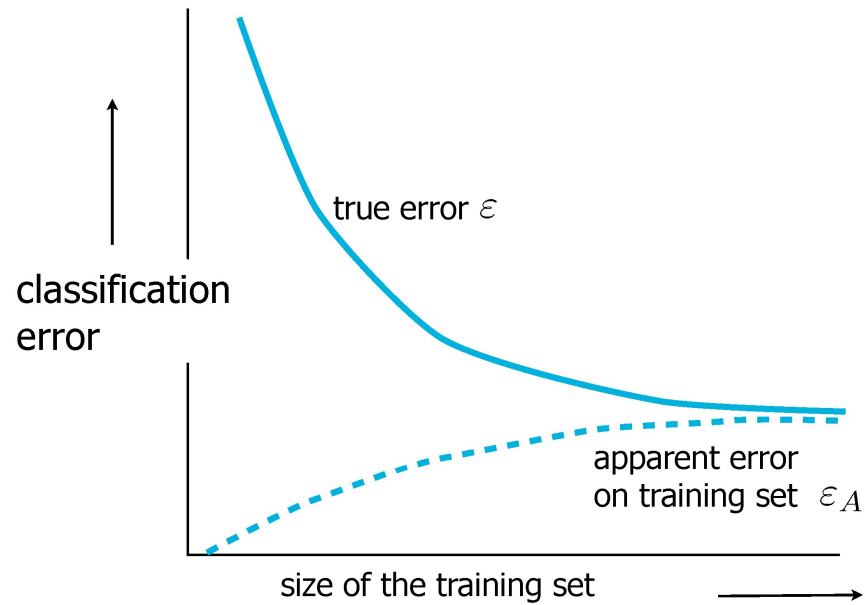
17

Error as a function of size of training set:



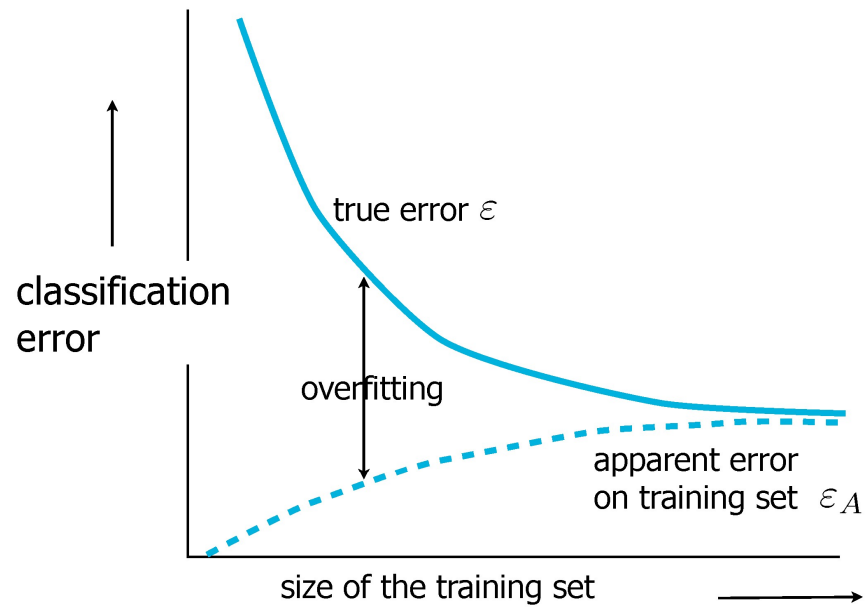
18

Error as a function of size of training set:



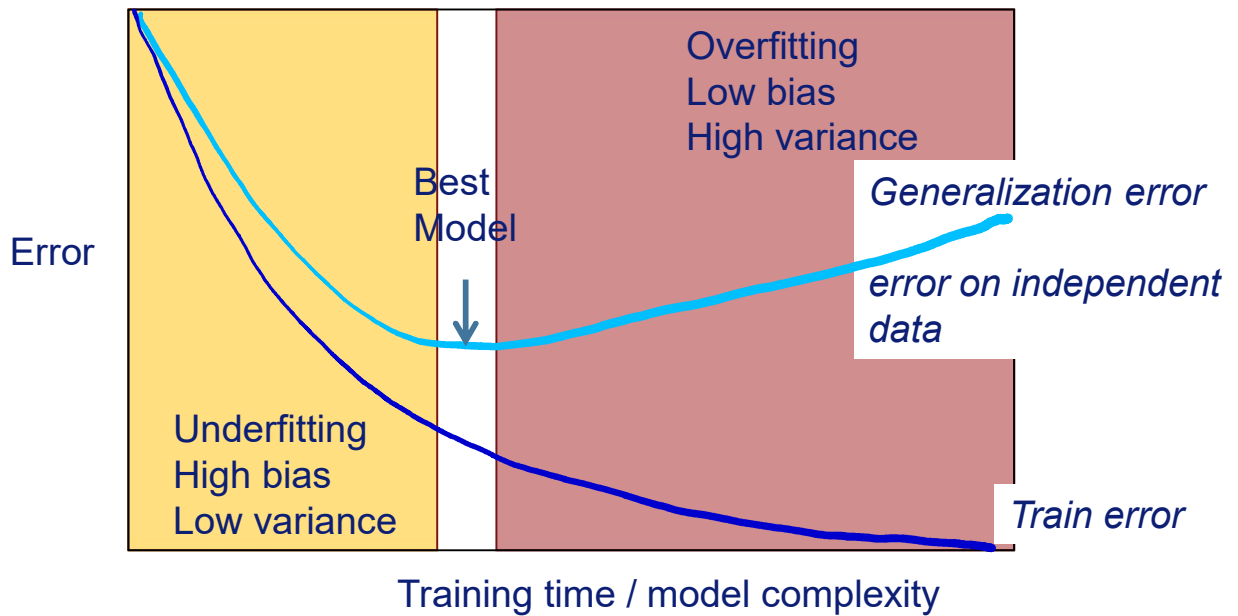
19

Error as a function of size of training set:



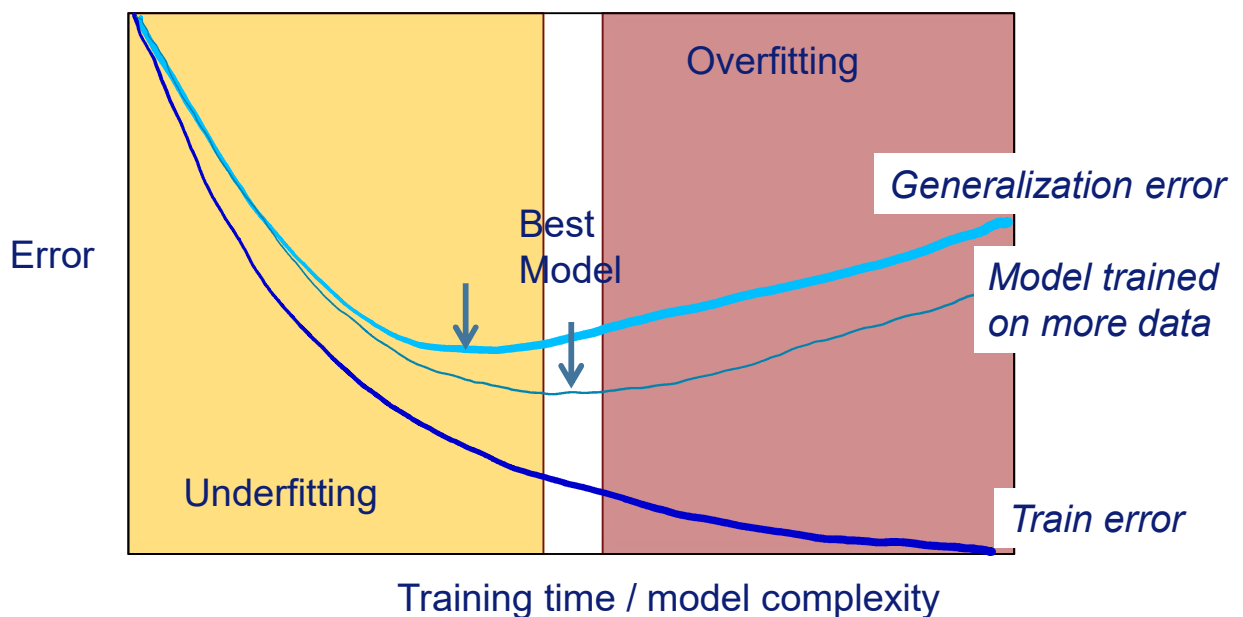
20

Error as a function of model complexity



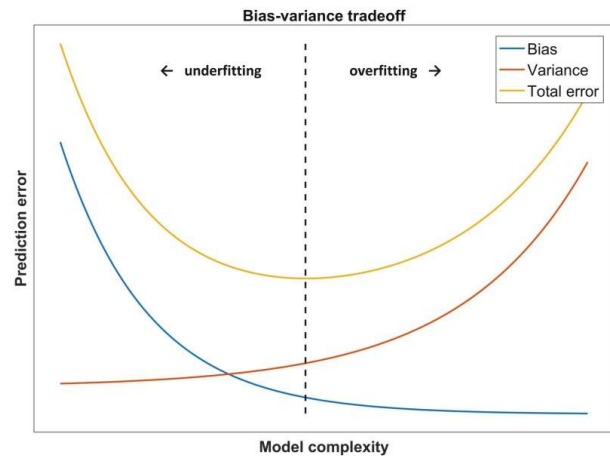
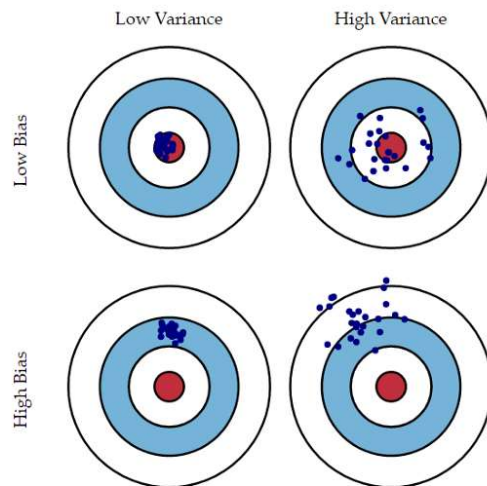
21

Error as a function of model complexity



22

Bias & variance



The aim with generalization is to reduce variance by increasing bias (hopefully) just a little

23

Regularization - Strategies to improve generalization

- Lecture 1 / Deep learning book, Chapter 5:
 - "**Regularization** is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error"
 - Aim: Move from overfitting regime to well-specified model
 - Or: Trade a little bias for large decrease in variance

24

Strategies to improve generalization

- First choice: Add more data!
- Constraints on the model (limit parameter values)
 - Extra terms in objective function (soft constraints on parameter values)
 - Sometimes designed to encode prior knowledge about the data
 - Often encodes a general preference for "simpler" models
- Data augmentation (adding "fake" data)
- Combining different models (ensembling)
 - Dropout

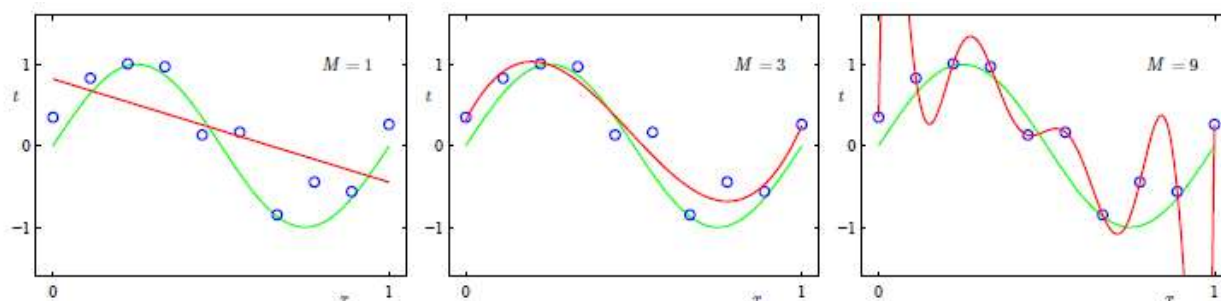
25

Constraints on parameters: Keeping weights small.

26

Keeping weights small. Why?

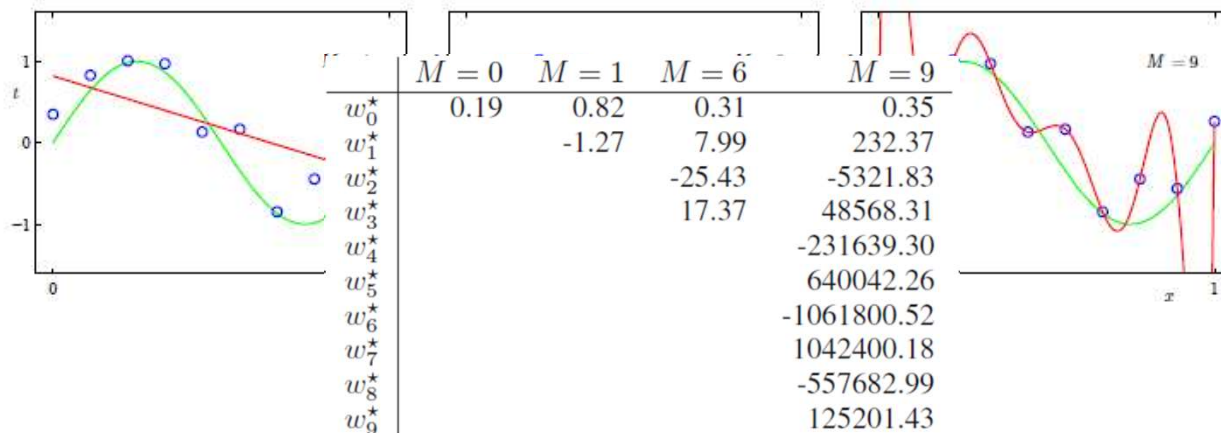
- Remember this one?



27

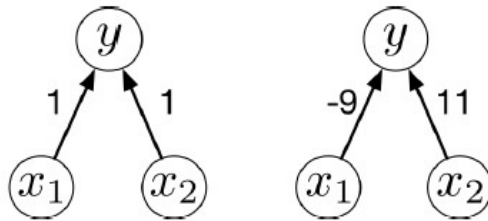
Keeping weights small. Why?

- Remember this one?



28

Keeping weights small. Why?



- Suppose x_1 and x_2 are the same
- These two networks give the same predictions.
- Small changes in x_1 and x_2 will give very different predictions
- The first network's predictions are more stable with respect to changes in input data

29

Keeping weights small. Why?

- Mostly, we want weights to be *similar*
- With smaller variations in weights, the model has less flexibility
- Allow varying weights only if it is really necessary to describe the data
- Fit the real correlations, not the spurious ones (if those are smaller)

30

Keep weights small. How?

- Add a penalty to the loss
- If L2 penalty: "weight decay"

$$\tilde{J}(w; X, y) = \frac{\alpha}{2} w^\top w + J(w; X, y), \quad (7.2)$$

Loss

L2 norm

- With derivative

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y). \quad (7.3)$$

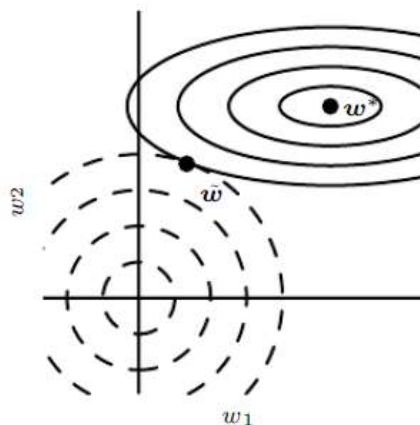
- The weight update becomes

$$w \leftarrow w - \epsilon (\alpha w + \nabla_w J(w; X, y)). \quad (7.4)$$

Weights are shrinking!

31

The geometric explanation



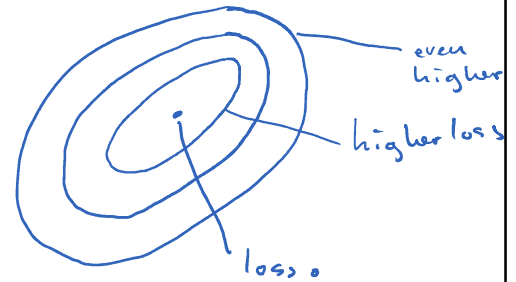
- Deep learning book, Fig 7.1, page 233
- What do they mean with this figure?

32

Assume the loss is a quadratic function*



In 1 dimension

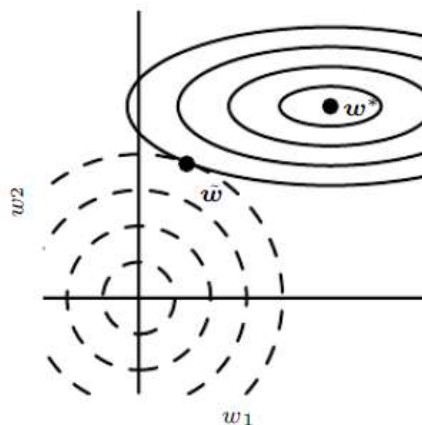


In 2 dimensions

* The figure shows quadratic loss, but this works for other shapes as well

33

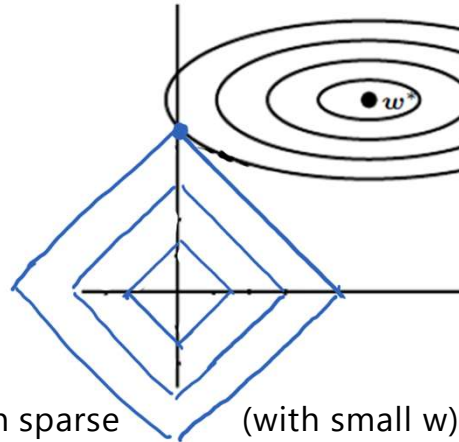
The geometric explanation



- Given the curve (ellipse) of equal loss according to the unregularized objective function, where is the point with minimum $\|w\|_2$?

34

What about L1 penalty?



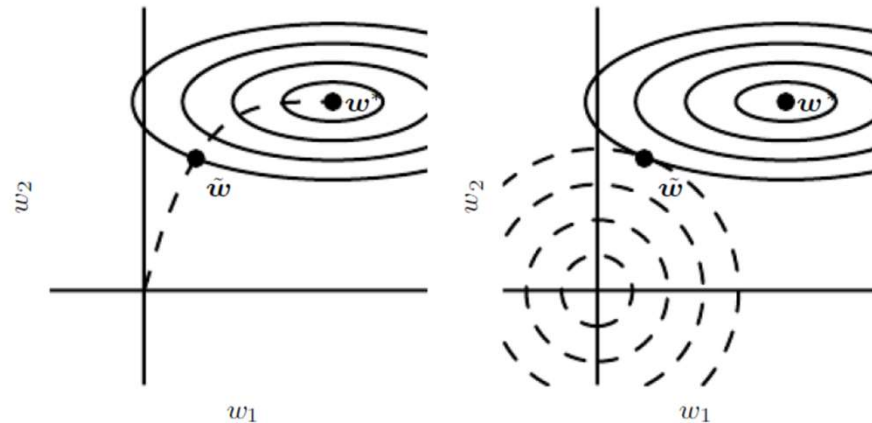
- L1 makes the solution sparse (with small w)
- Also used for feature selection.
- In Networks: Learns which neurons need to be connected. Faster inference.

35

Other ways to keep weights small?

36

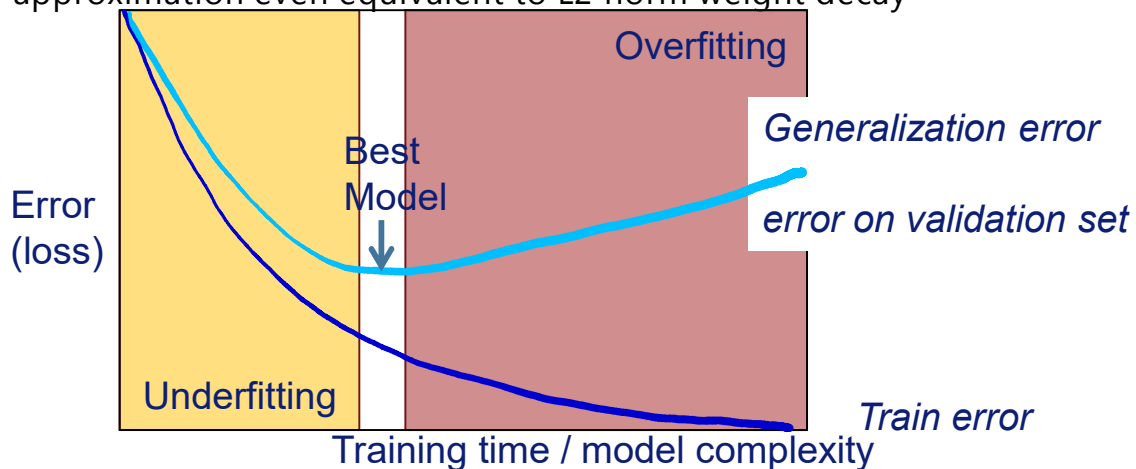
Gradient descent from initialization close to zero:



37

In networks: Early stopping

- During training, keep track of loss on validation set
- Stop when validation loss increases
- This is also a way to keep weights small; with quadratic error function approximation even equivalent to L2 norm weight decay



38

So, for early stopping

- Divide your data in three parts
 - A **test set** you donot use until the very final evaluation of your model of choice
 - A **training set** in which you optimize the weights by backprop
 - A **validation set** to track generalization error
- How to choose hyperparameters? (# layers, nodes, etc)
 - Typically, by performance on validation set
 - But need to be careful, may be overfitting to validation set

39

Further constraints or penalties on the parameters

- Can some parameters be shared (so they are equal) or tied (so they are similar – their difference is penalized)?
- This is what convolutional neural networks do!

40

Strategies to improve generalization

- First choice: Add more data!
- Constraints on the model (limit parameter values)
 - Extra terms in objective function (soft constraints on parameter values)
 - Sometimes designed to encode prior knowledge about the data
 - Often encodes a general preference for "simpler" models
- Data augmentation (adding "fake" data)
- Combining different models
 - Dropout

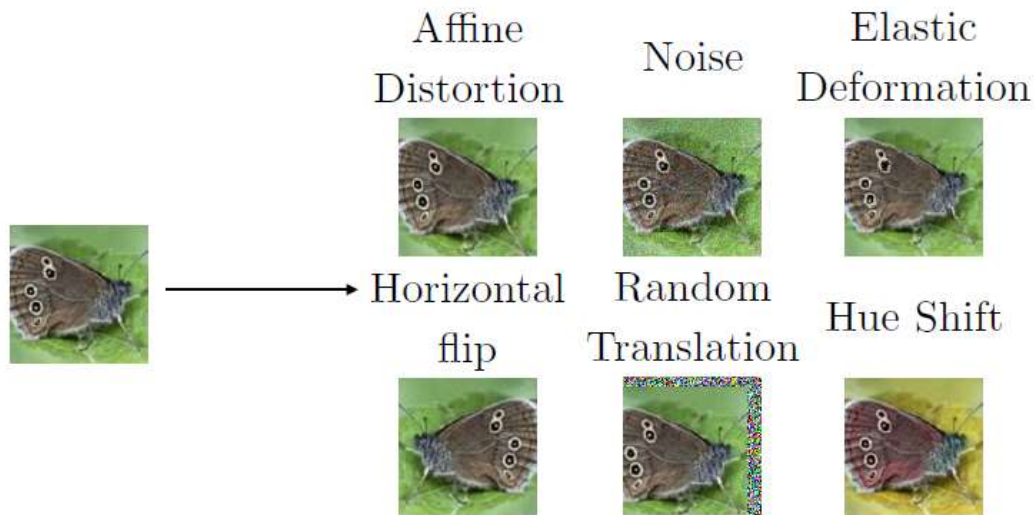
41

Data augmentation

- Best way to improve generalization is add more training data
- Maybe we can synthesize new samples?
- Fake data!

42

Data augmentation for images



- Can you think of other ways to augment the image?
- What do these transformations have in common?



43

Data augmentation

- Can help a lot even if changes to the data are small
- Should be safe as long as new data is "plausible"
- Makes the model invariant to the augmentation transformations
 - Do we want this?
 - For instance, if color is important for recognition, be careful with color augmentation
 - Easy way to teach the model to cope with variations in lighting, zoom, viewing angle, etc
- Might still help also if new data is highly implausible
- For instance, simply averaging pairs of input images works quite well in some circumstances

44

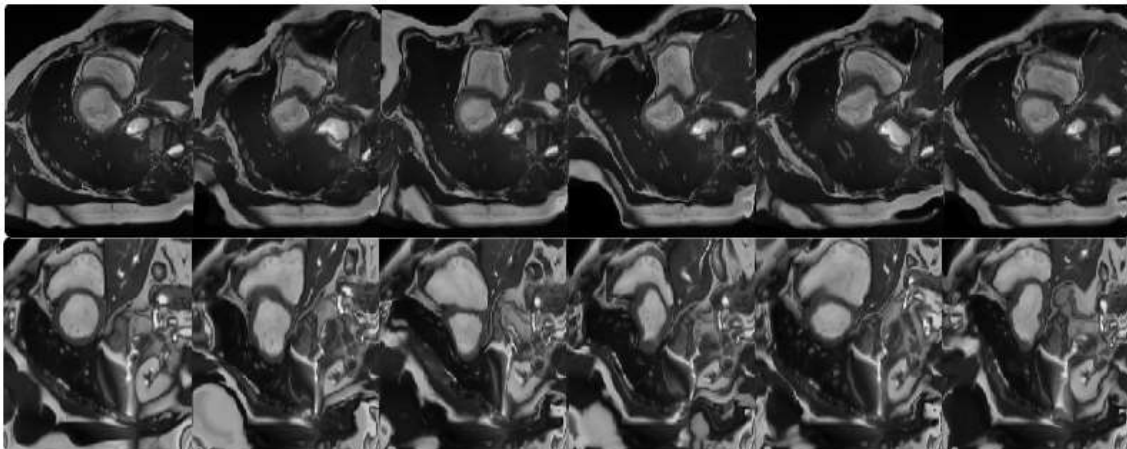
Highly implausible images can be helpful for classification

$0.4 \times \text{data}[1]:$		cat: 1.0
$+ 0.6 \times \text{data}[2]:$		dog: 1.0
$=$	mixup:	cat: 0.4 dog: 0.6

45

Highly implausible deformations can be helpful for segmentation

- Chaitanya et al (IPMI 2019/MEDIA 2020) searched the elastic deformation augmentation parameters that improved heart segmentation maximally:

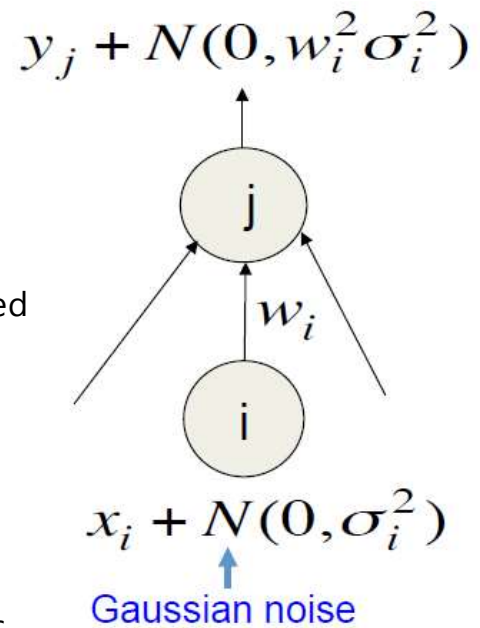


Input Image | Images generated by $G_V \rightarrow$

46

Injecting noise

- Most common:
 - On the inputs = data augmentation
 - Also: adding noise to input and minimizing squared error will result in minimizing squared weights, similar to L2 norm
 - On the weights
 - On the output: label smoothing
 - Encodes that there is some uncertainty in labels
 - Avoids learning of "unreachable targets" 0 and 1 in classification



47

Strategies to improve generalization

- First choice: Add more data!
- Constraints on the model (limit parameter values)
 - Extra terms in objective function (soft constraints on parameter values)
 - Sometimes designed to encode prior knowledge about the data
 - Often encodes a general preference for "simpler" models
- Data augmentation (adding "fake" data)
- Combining different models
 - Dropout

48

Ensembles: combining multiple networks

- As in the lecture on combining learners, train multiple networks and combine (eg average) their outputs
- Why does this work?
 - Different networks make different errors
 - Combining their predictions reduces variance
- The networks to combine could be:
 - Multiple runs of the exact same network with different random seed (Yes, they can be quite different!)
 - The same network trained on different subsets of data
 - Networks with different architectures, loss functions
 - Special case: dropout

49

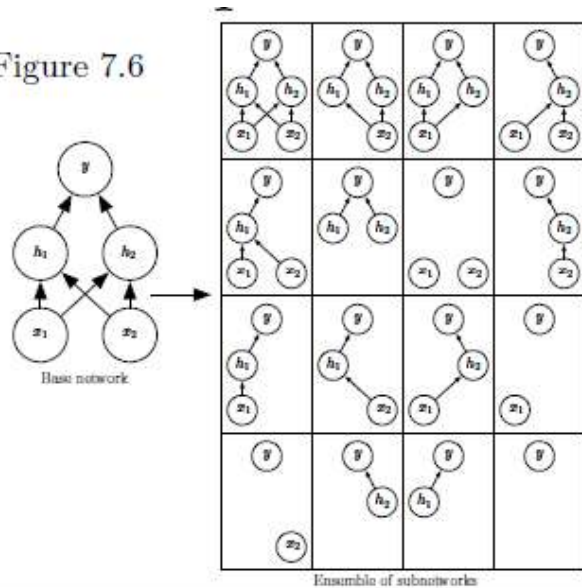
Ensembles: combining multiple networks

- Model averaging is an easy and robust way to improve generalization
- Drawbacks?
 - Neural network training time typically long
 - training 10 networks takes 10 times longer
- Efficient way of ensembling: dropout

50

Dropout: Randomly sample subnetworks

Figure 7.6



- For each sample, randomly draw a mask to remove some nodes
- Up-scale the rest of the values to keep the average activation at about the same level.
- Forward & Backpropagate as usual
- Note: in this example many subnetworks have no input or no connection between input and output. In practice, we always have larger inputs and larger hidden layers and this is no longer an issue

51

Dropout

- Can be seen as efficient way of bagging
- Quite successful as regularizer in deep networks
- May be less effective for smaller datasets? (the deep learning book, p265)
- Parameters to tune – may need different probability of dropout in different layers
- It reduces model capacity – may require increasing the number of hidden nodes to reach low error rates
- Takes more time to train
- Variants: dropping input channels, connections, dropping complete featuremaps, dropping complete layers, multiplying by random values instead of selecting

52

Dropout – more than bagging?

- In regular bagging (ensembling) approaches, the individual learners are independent and trained independently
- This works best if individual learners are more independent
- In dropout, individual networks not independent, share many parameters
- -> less averaging out of mistakes?
- But, in fixed network, hidden units may learn to adapt to other units (co-adaptation" – they learn to correct small errors that other units make. Less generalizable.
- With dropout, each unit has to work well with many possible combinations of other units. Is therefore more likely to learn something useful.
- Also, sharing weights between models is a strong regularizer. Instead of keeping weights small (L1/L2 norm penalties) pull weight towards a value that is probably useful

53

Dropout is a regularizer during training. What to do at inference time?

- Two options:
 - Apply the "mean" model (scale weights if you have not done so during training)
 - Apply dropout during inference as well.
 - Why would you do this?
 - It gives an estimate of the uncertainty in the labels

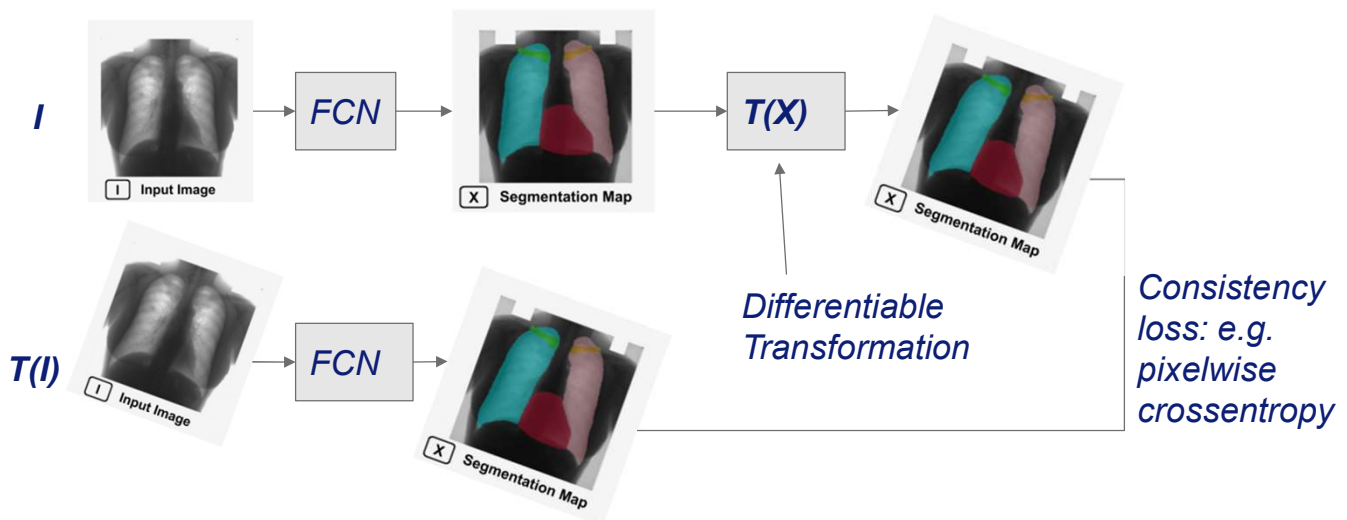
54

Advanced tricks to improve generalization:

- Multi-task optimization: One network learns to solve multiple tasks at the same time (or alternately)
- Can be useful if tasks share common information
- For example: Image classification and image reconstruction
- Can also be used in the semi-supervised learning setting
 - For images without labels, optimize based on reconstruction objective
 - For images with labels, optimize reconstruction and classification

55

Consistency loss can be used both with or without labels (supervised and unsupervised)



Bortsova et al, 2019

56

- With material from
- <http://www.deeplearningbook.org/>
- Roger Grosse http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/
- David Tax and Marco Loog
- Geoffrey Hinton