DNNs Best Practices In the State-of-the-art Techniques

Patrik Kenfack
p.kenfack@innopolis.university

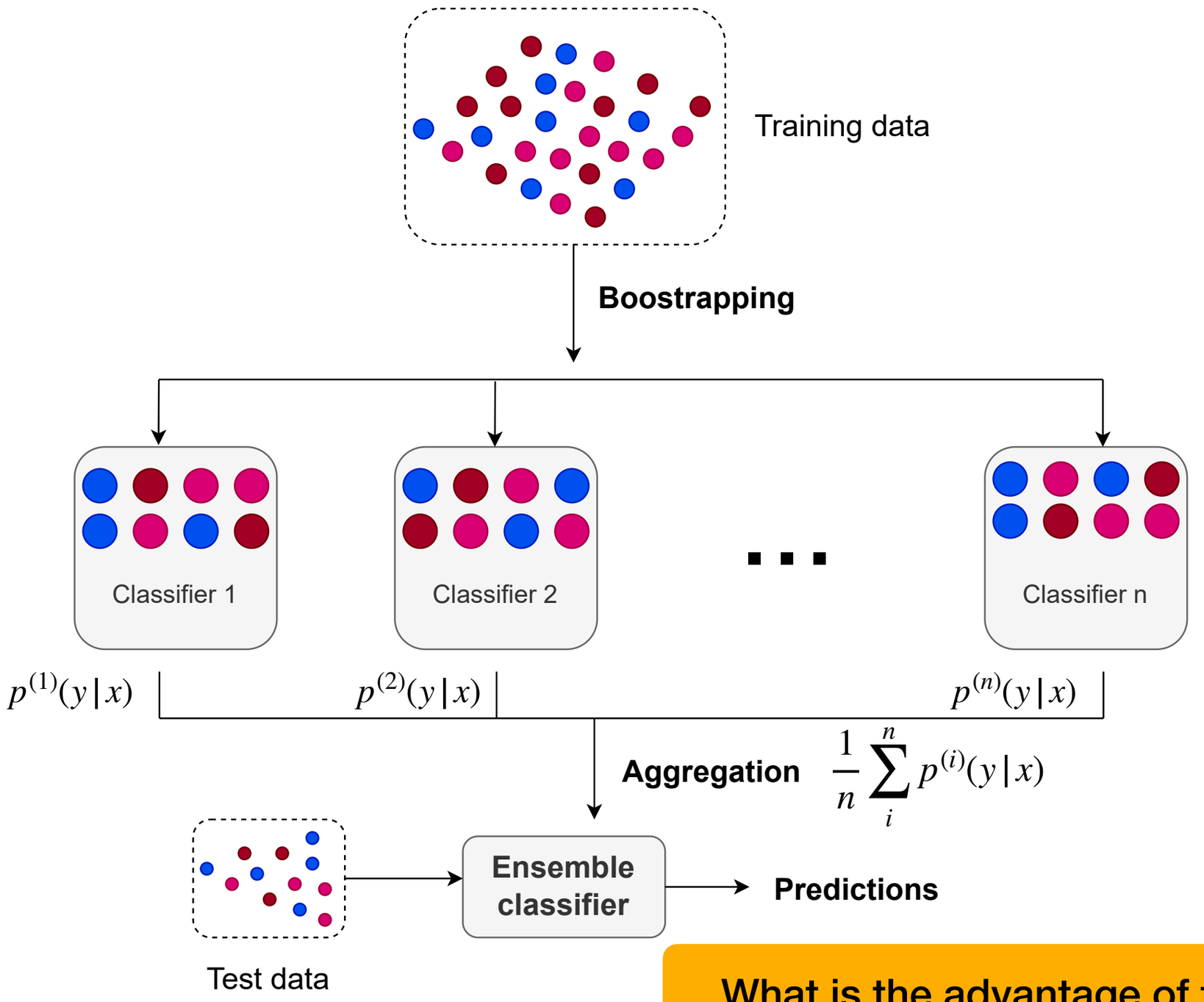
Objectives

- Dropout
- Batch Normalization
- Early Stopping
- Learning Rate scheduling
- Gradient Clipping
- Optimizers
- Data Augmentation
- Using Pre-trained layers

Dropout

Dropout

We need first to know Bagging



What is the advantage of this approach?

Dropout

Bagging is an ensemble method that averages the predictions of many high-variance learners so the collective prediction has ***lower variance***.

How?

Let the predictions of each learner $\hat{y}^{(i)} = p^{(i)}(y \mid x)$ be a random variable and

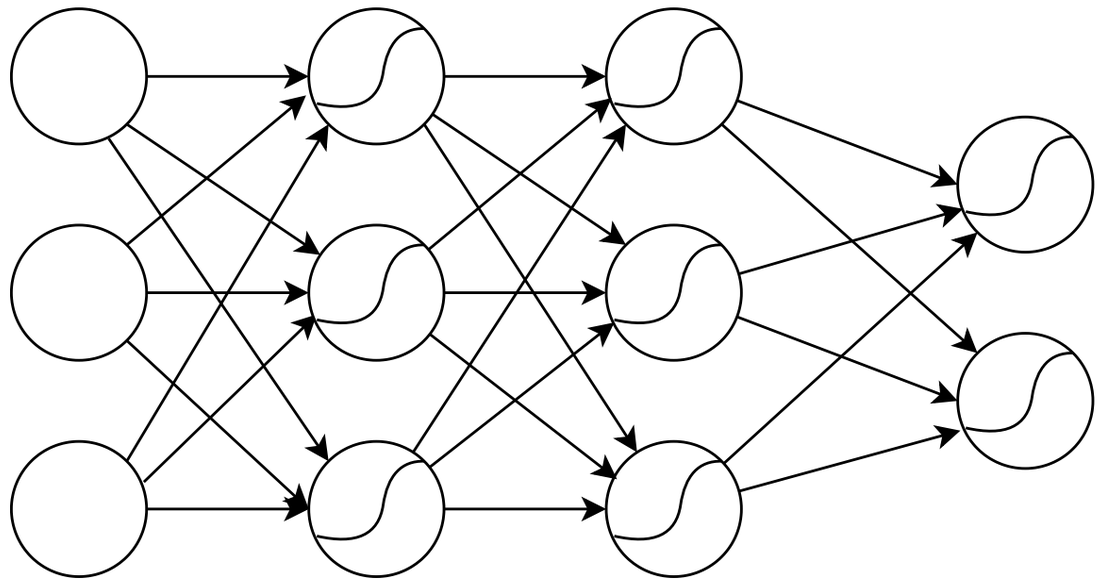
all of them are i.i.d. with variance σ^2 so the variance of their mean:

$$\mathrm{Var}\left(\frac{1}{n}\sum_i \hat{y}^{(i)}\right) = \frac{1}{n^2}\mathrm{Var}\left(\sum_i \hat{y}^{(i)}\right) = \frac{1}{n^2}\sum_i \mathrm{Var}\left(\hat{y}^{(i)}\right) = \frac{1}{n^2}\sum_i \sigma^2 = \frac{1}{n^2}\cdot n \cdot \sigma^2 = \frac{\sigma^2}{n}$$

So, the variance of the mean is less, which it means less overfitting.

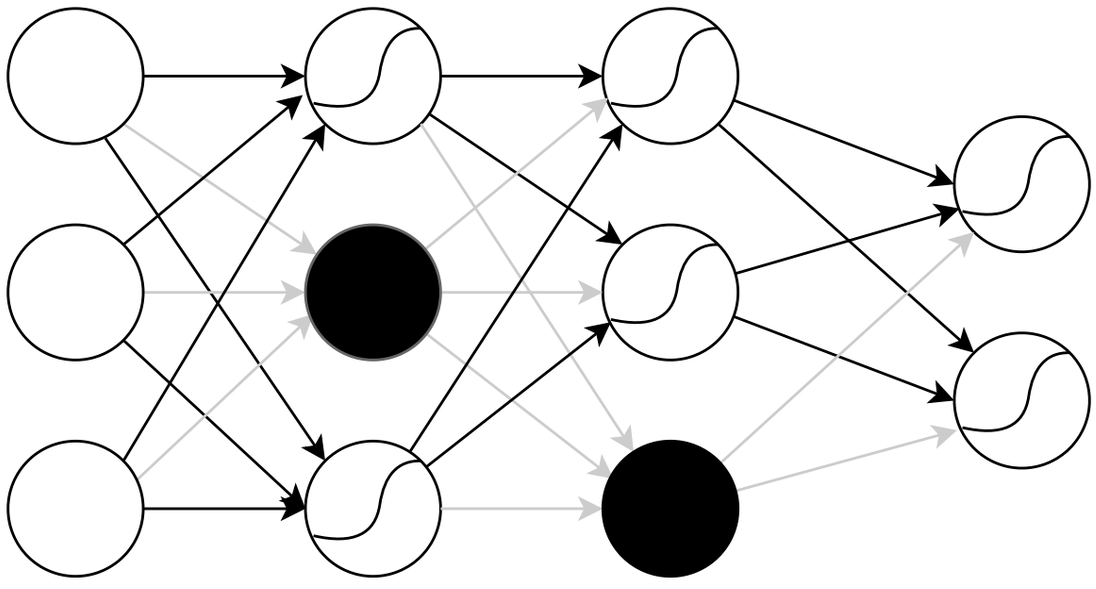
Dropout

- Dropout aims to approximate bagging process in neural networks.
- It randomly removes nodes from the neural network
  - With a probability p (e.g., 0.5) each unit is set to 0, i.e, stop their forward propagation to the next layers
  - So all possible produced networks are theoretically enough to make prediction (as an individual model)



Dropout

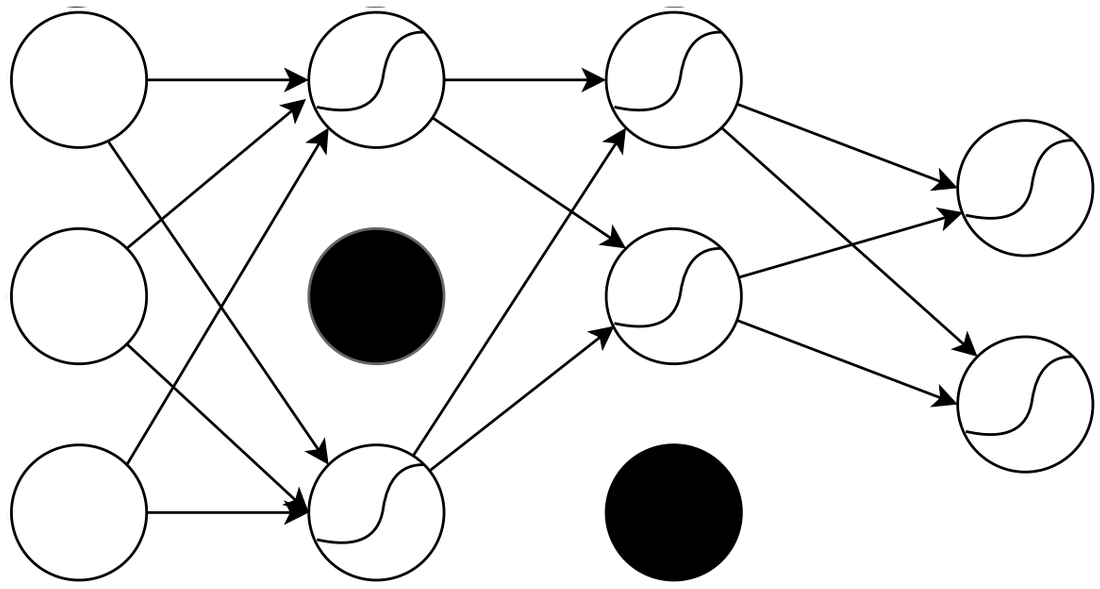
- Dropout aims to approximate bagging process in neural networks.
- It randomly removes nodes from the neural network
  - With a probability p (e.g., 0.5) each unit is set to 0, i.e, stop their forward propagation to the next layers



Dropout

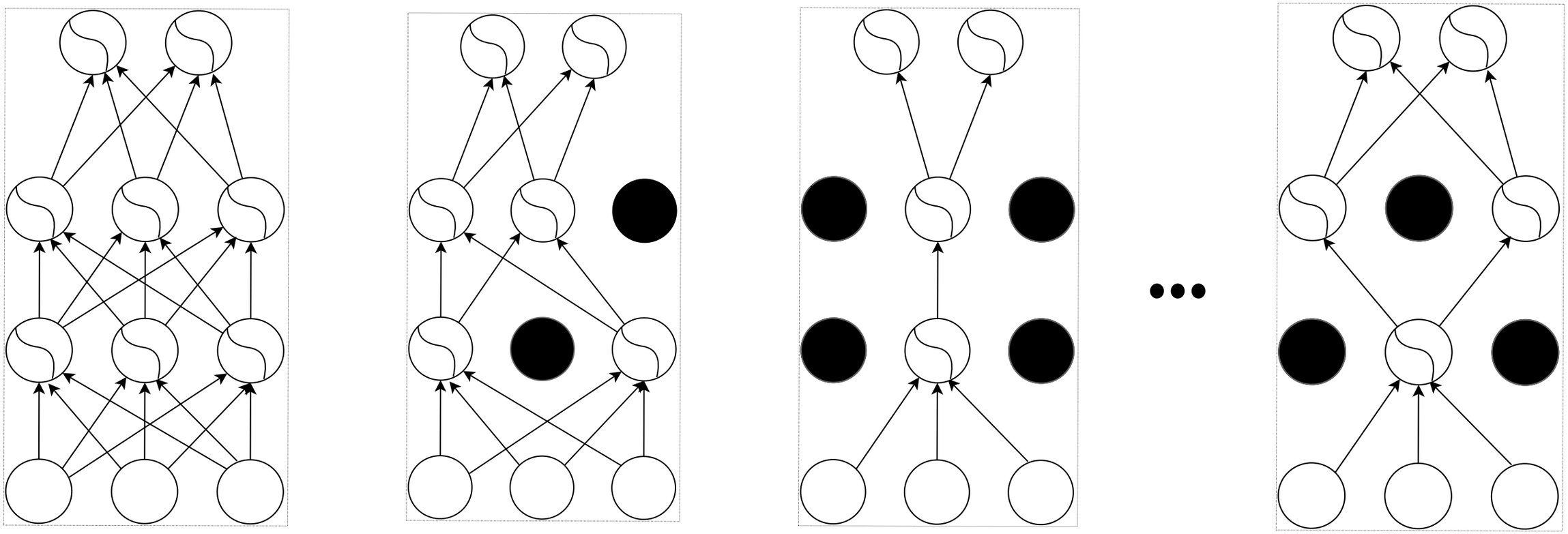
- Dropout aims to approximate bagging process in neural networks.
- It randomly removes nodes from the neural network
  - With a probability p (e.g., 0.5) each unit is set to 0, i.e, stop their forward propagation to the next layers

The remaining units forms a subnet (individual model)



Dropout

- In each forward pass, each (hidden) unit is randomly dropped with a probability p

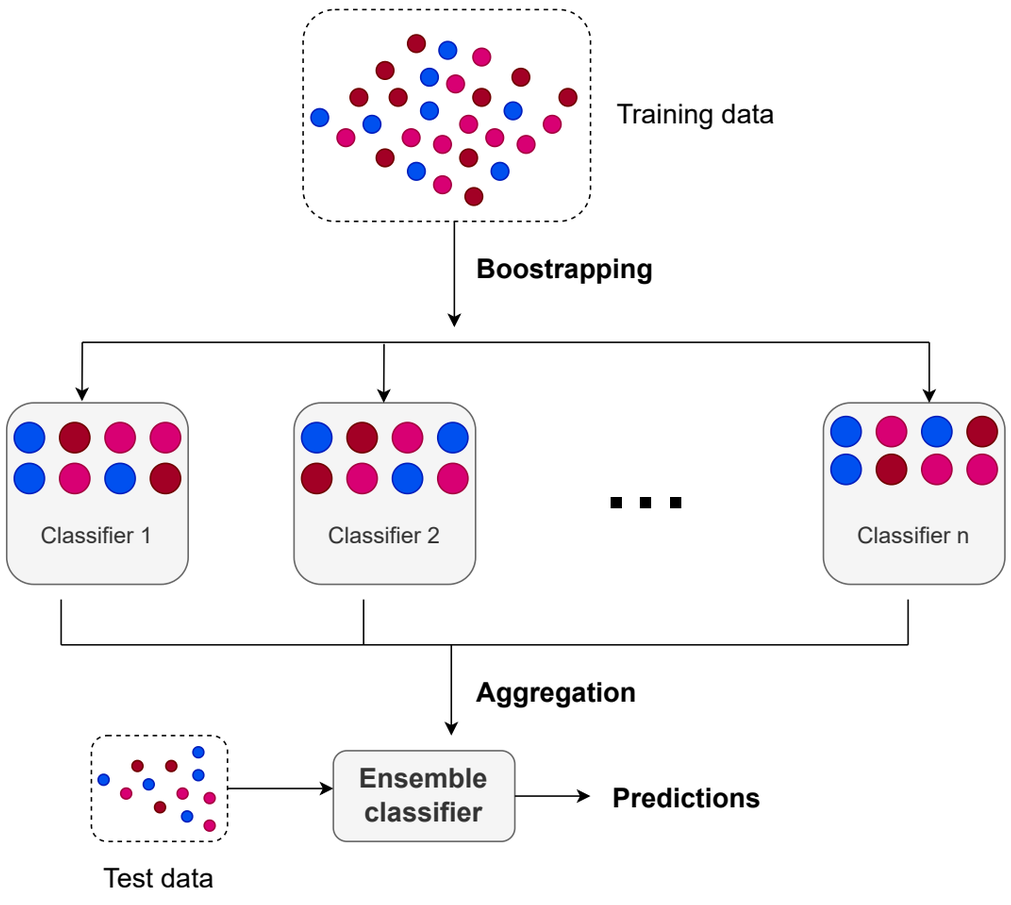


There can be an exponentially growing number of subnets

Dropout

Bagging

$$\frac{1}{n}\sum_{i}^{n} p^{(i)}(y \mid x)$$



Dropout

Intractable

$$\sum_{\mu} p(\mu) \cdot p^{(i)}(y \mid x)$$

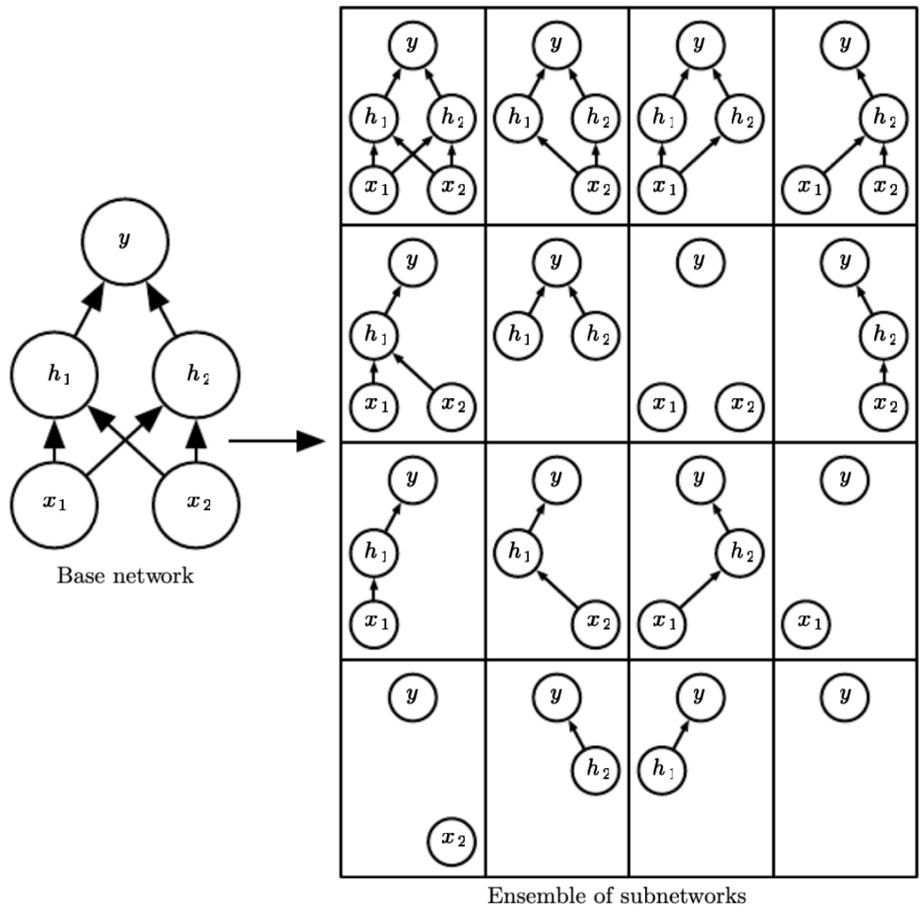


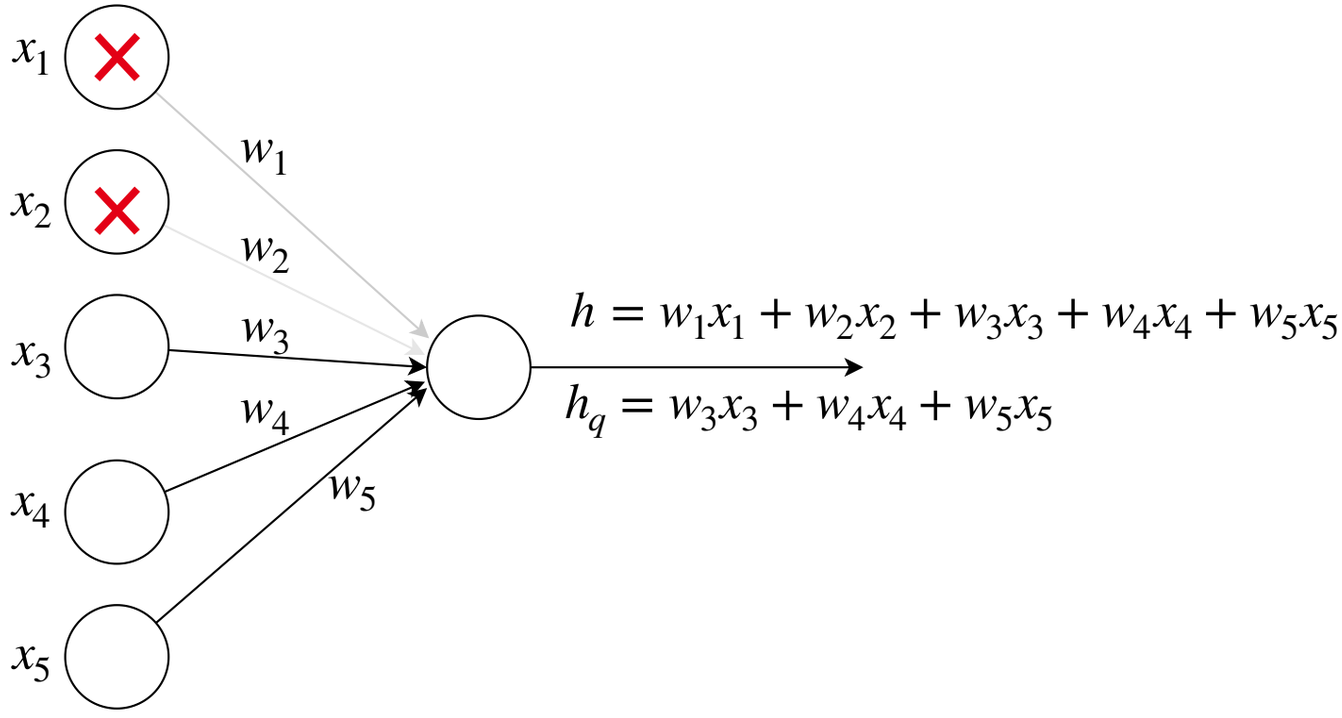
Figure 7.6 of this book

Dropout

Why does dropout work for regularizsation?

- In each forward pass units are randomly dropped with a probability p
- Thus, each unit cannot rely on any of its inputs, forcing the weights to spread out (shrink). This is similar to l_2 regularization

With the probability p of dropping a unit in a layer, what is the expected number of units remaining in a layer after the dropout ?



Dropout

Notes

- Dropout is a type of regularization of the network to reduce the overfitting, so it is usually done when the network is very big, too long training or small amount of training data.

- Usually the dropping probability p is 0.5 or lower. Researchers recommend values between 0.2 and 0.5

- Only applied during the training.

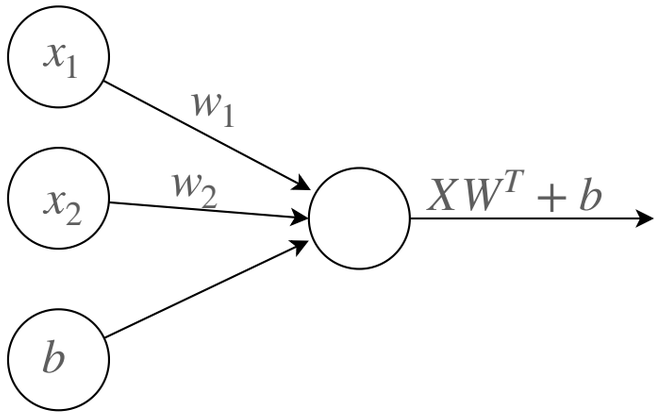
- Dropout is commonly used after dense (fully-connected) layers and rarely used after convolutional layers, even when it is done there, it has low dropping probability p.

Read more about dropout
section 7.12 of this book

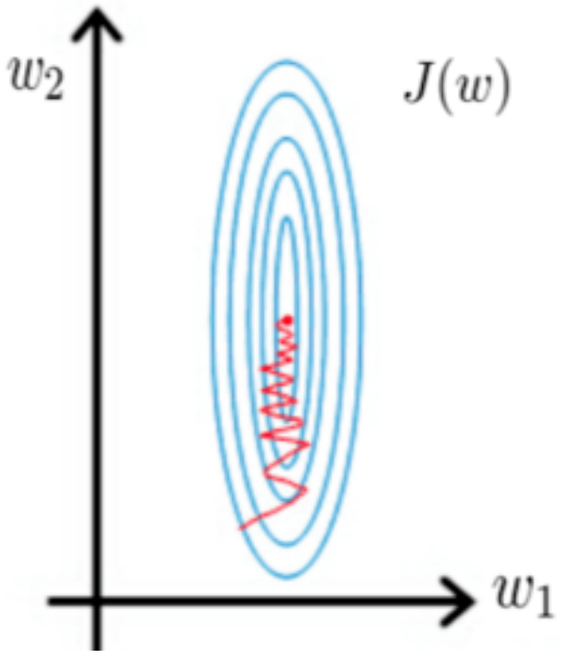
Batch Normalization

Batch Normalization

- Recall the impact of data normalisation or standardization on the gradient descent algorithm

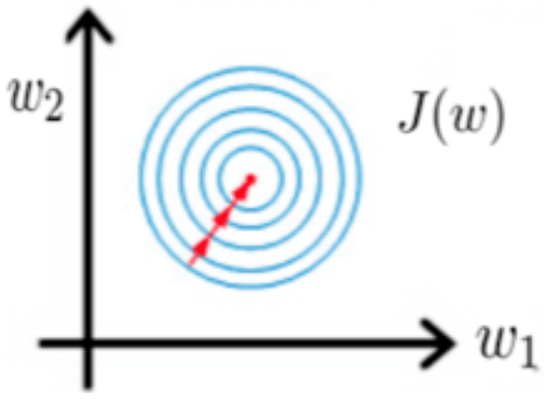


$x_1 >> x_2$



Feature Scaling

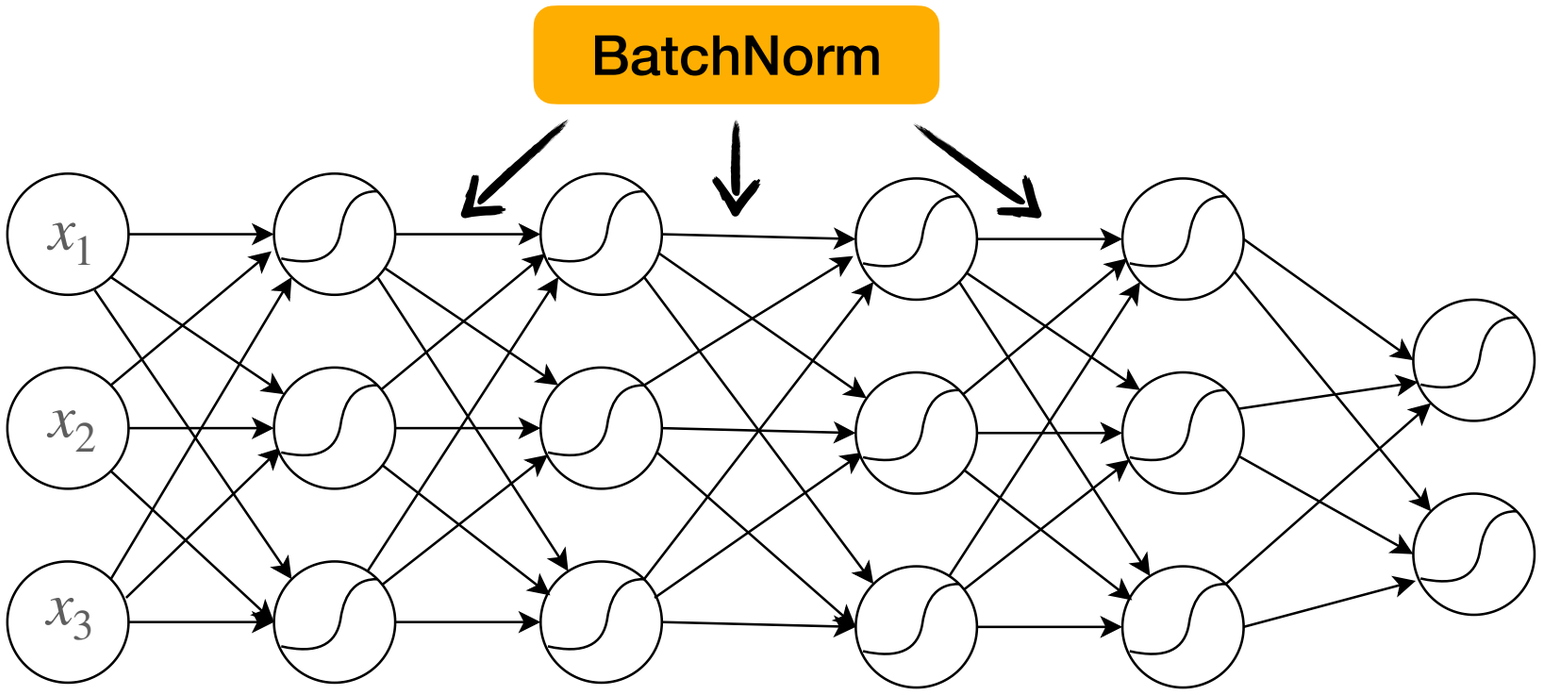
$0 \leq x_1 \leq 1$
$0 \leq x_2 \leq 1$



Batch Normalization

- During gradient descent, weights are simultaneously updated assuming weights of layers others remains fixed, not true in practice.
  The distribution assumed by layers can change (**internal covariance shift**)

- Solution: For each mini-batch, Standardize the hidden unit values before feeding to the next layers



Batch Norm Layer

At training time

- Let Z be the mini-batch of activations for the layer to normalize

  $\mu = \frac{1}{m}\sum Z_i$ the mean actions of the each unit in the mini-batch

  $\sigma^2 = \frac{1}{m}\sum_i (Z_i - \mu)^2$

  The normalized values of the units is $Z' = \frac{Z - \mu}{\sqrt{\sigma^2 + \epsilon}}$

- To preserve the expressiveness of the model, Z' is replaced by
  $Z' = \gamma Z' + \beta$
  γ and β are learnable parameters. To allow any mean different from 0

Batch Norm Layer

At testing time

- At testing test we can evaluate the model one a single sample. Thus, we may be be able to have μ and σ^2
- μ and σ^2 may be replaced by **running averages** that were collected during training time.

Batch Normalization

Notes

- BatchNorm has a great effect in stabilizing the training process and reduce the number of epochs needed to converge.

- Order of Layers:

  Conv → Activation → MaxPool → BatchNormalization

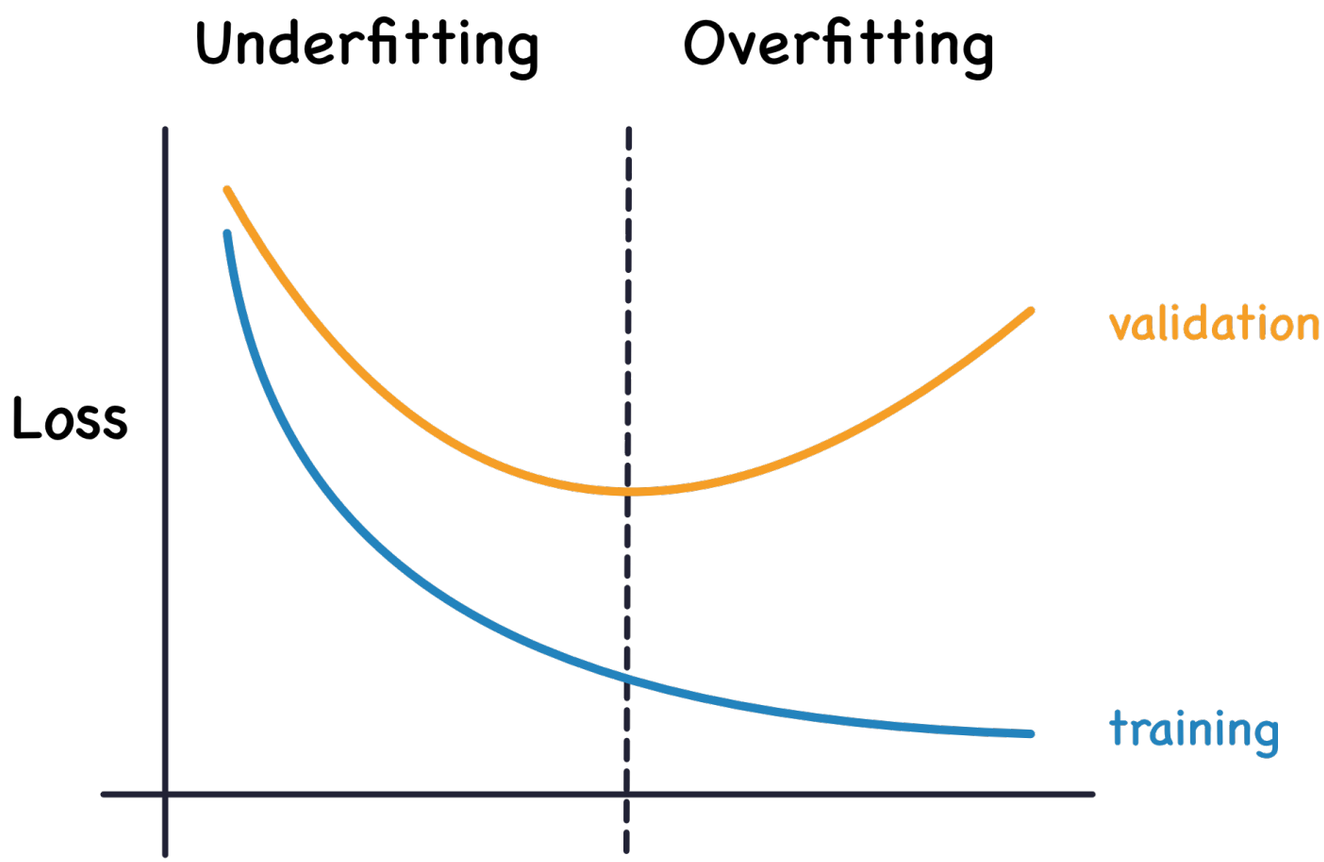
  Danse → Activation → BatchNormalization → Dropout

- Sometimes provide some regularization.

Read more about batch normalization section 7.8 of this book

Early Stopping

Early Stopping



Early Stopping

- When training a network with capacity to overfit the training data, the training set loss decrease steadily while the validation loss goes up and down.

- Early stopping allows to restore network parameters to the point in time where there was smallest validation loss.

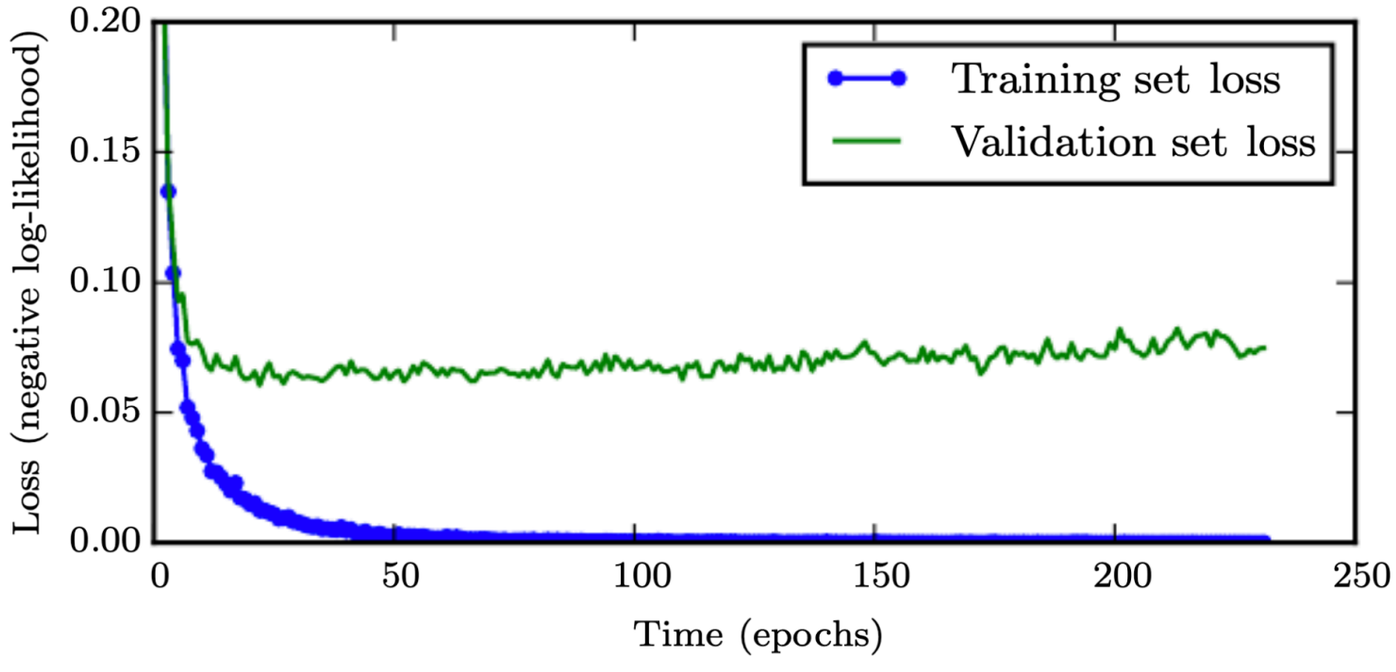


Figure 7.3
of this book

Intuition: Using the weights with lowest validation loss might result in better performance on the test set (better generalisation)

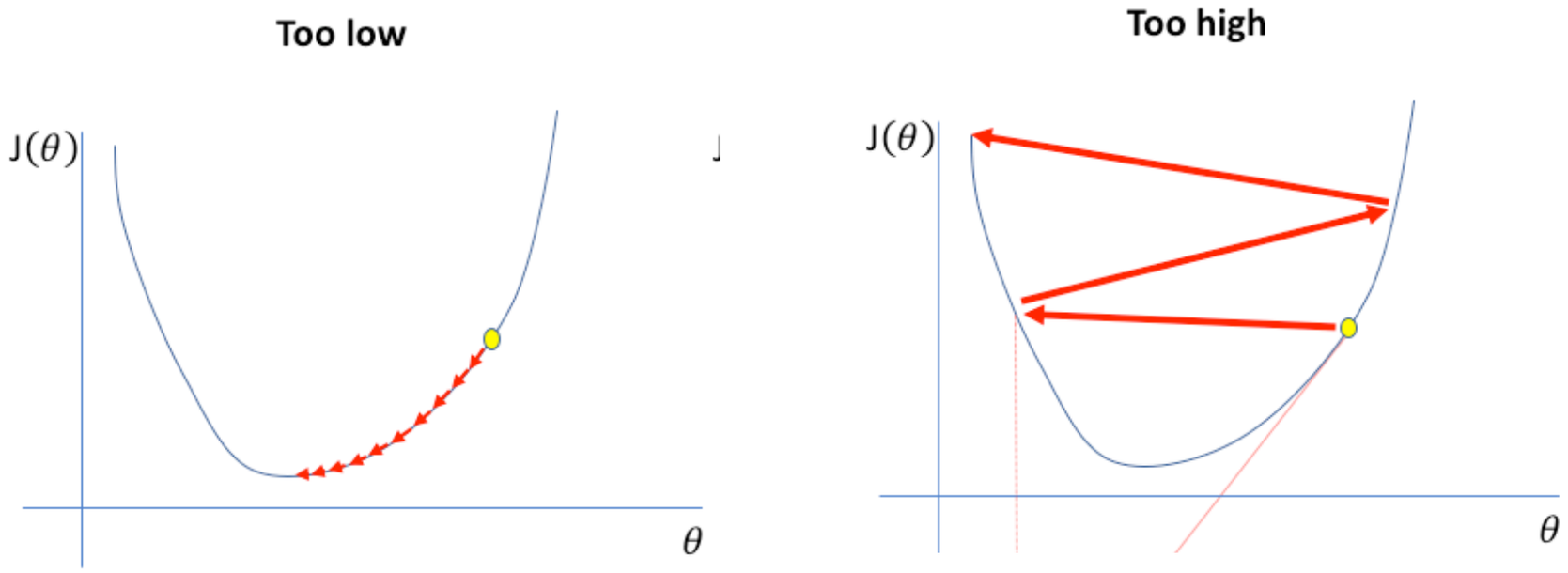
Early Stopping

- During the training:
  - Store (copy) the parameters that improve the validation loss
  - Stop the training when there is not improvement after some predefined number of epochs (called patience in Keras).
  - Return the stored parameters after the training
- Early stopping can be used find the number of steps needed to training a model, seen as hyperparamter.
- Commonly used form of regularization in DNNs

Learning Rate Scheduling

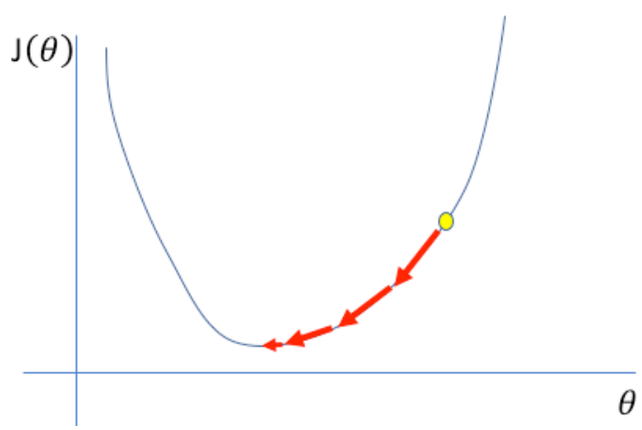
Learning Rate Scheduling

- The choice of the LR is very crucial in training DNNs during gradient descent
  - Small value of LR — the training may takes more time to converge
  - Lager value of LR — training unstability or divergence



Learning Rate Scheduling

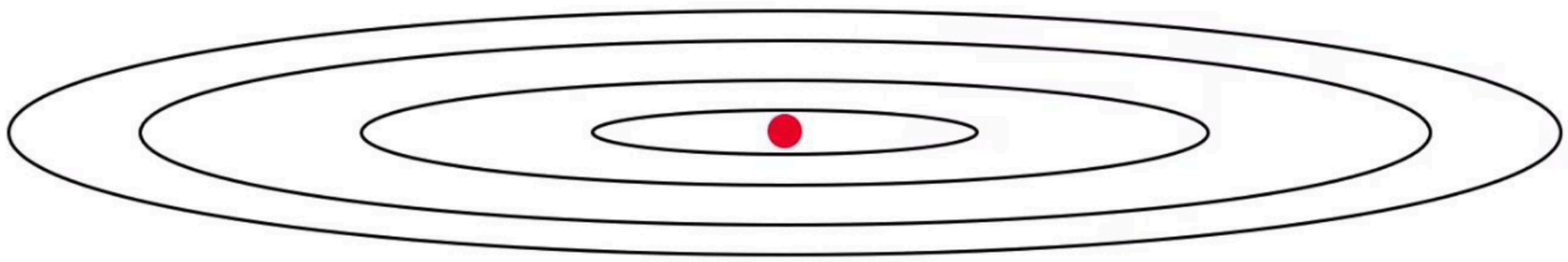
- Scheduling the learning rate is considered one of the best practices in training a DNN as an alternative of using fixed learning rate
    - Start with initial relatively large learning rate and keep reduce it (somehow) till the end of the training process
    - Update the learning rate small by a factor when the performance of the model plateaus
- Allows large update at the beginning and fine-tune the updates progressively



Optimizers

Optimizers

- The traditional optimizer for DNN is SGD which is updating the weight with a constant learning rate.
- This is not appropriate if some weights needs bigger steps than the others to reach the optimal point.
- The step size is computed by the learning rate and the gradient value. As we are using mini-batches GD, the gradient may not always be as sharp, however, its moving average will show how much we need to reach the minimal point



Optimizers

- Most of the adaptive optimizers try to estimate the moving average of the gradient for each weight and update it based on that.
- The recommendation of which optimizers to use:
  - Adam → RMSProp → SGD with momentum

Gradient clipping

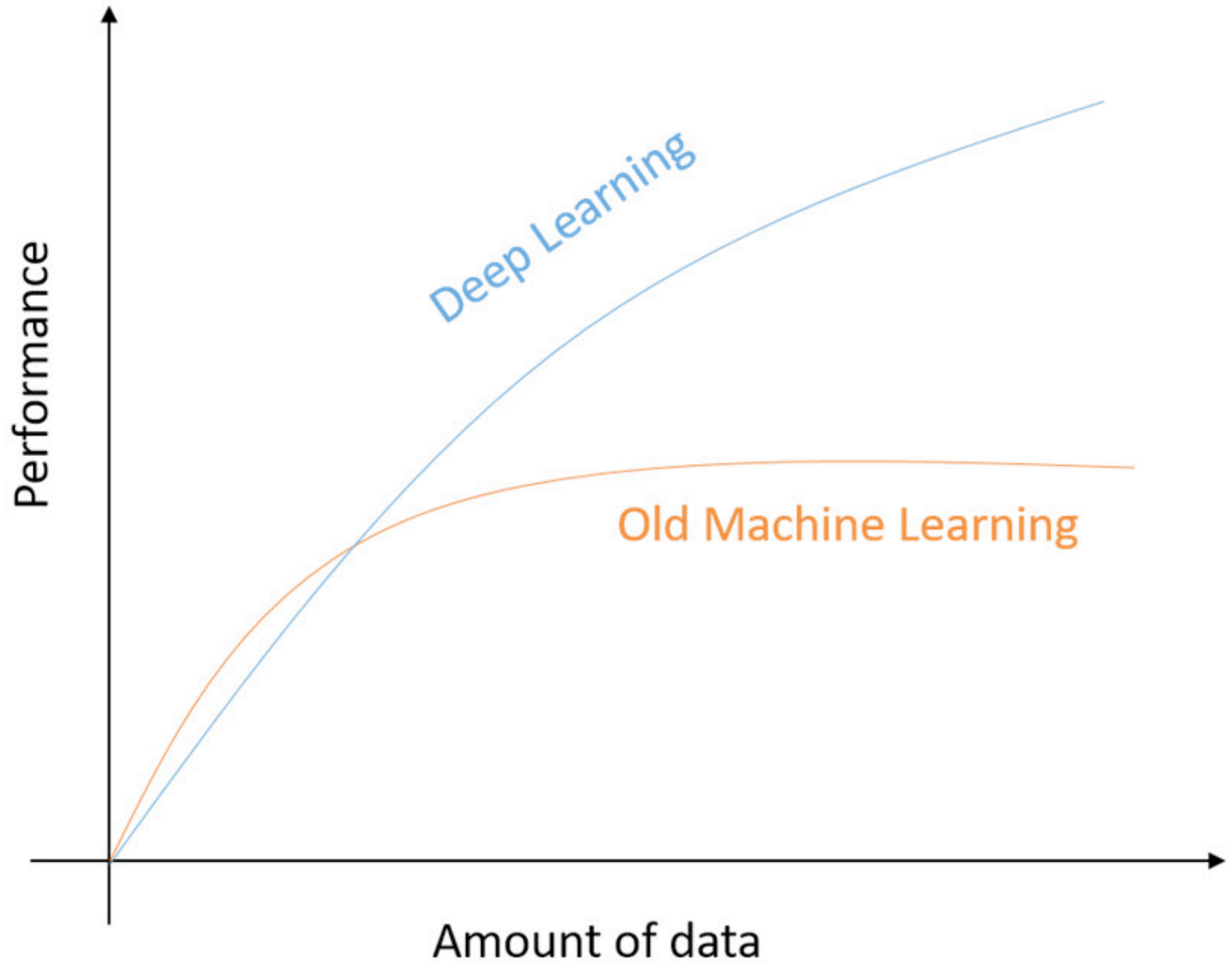
Gradient clipping

- A technique used to prevent the gradient from exploding and going out of control in deep architecture—specially in recurrent neural nets.
- Two approaches to do so:
  - Norm Clipping: $\delta = \delta \frac{C}{\|\delta\|_2}$ clip the gradient if its l2 norm exceed C
  - Value Clipping: $\text{clip}(\delta, -C, C)$ if the value at any index in the gradient tensor is not the range $[-C, C]$

Read more section 8.2.4 of this book

Data augmentation

Data augmentation



Data augmentation

- Augmentation is the best way to make the models generalize better by applying random (appropriate) transformations to the original dataset. It can adds some regularization to avoid overfitting

Original image

Horizontal flipping

Vertical flipping

Data augmentation

- Augmentation is the best way to make the models generalize better by applying random (appropriate) transformations to the original dataset. It can adds some regularization to avoid overfitting

Original image

22.5 rotation

-22.5 rotation

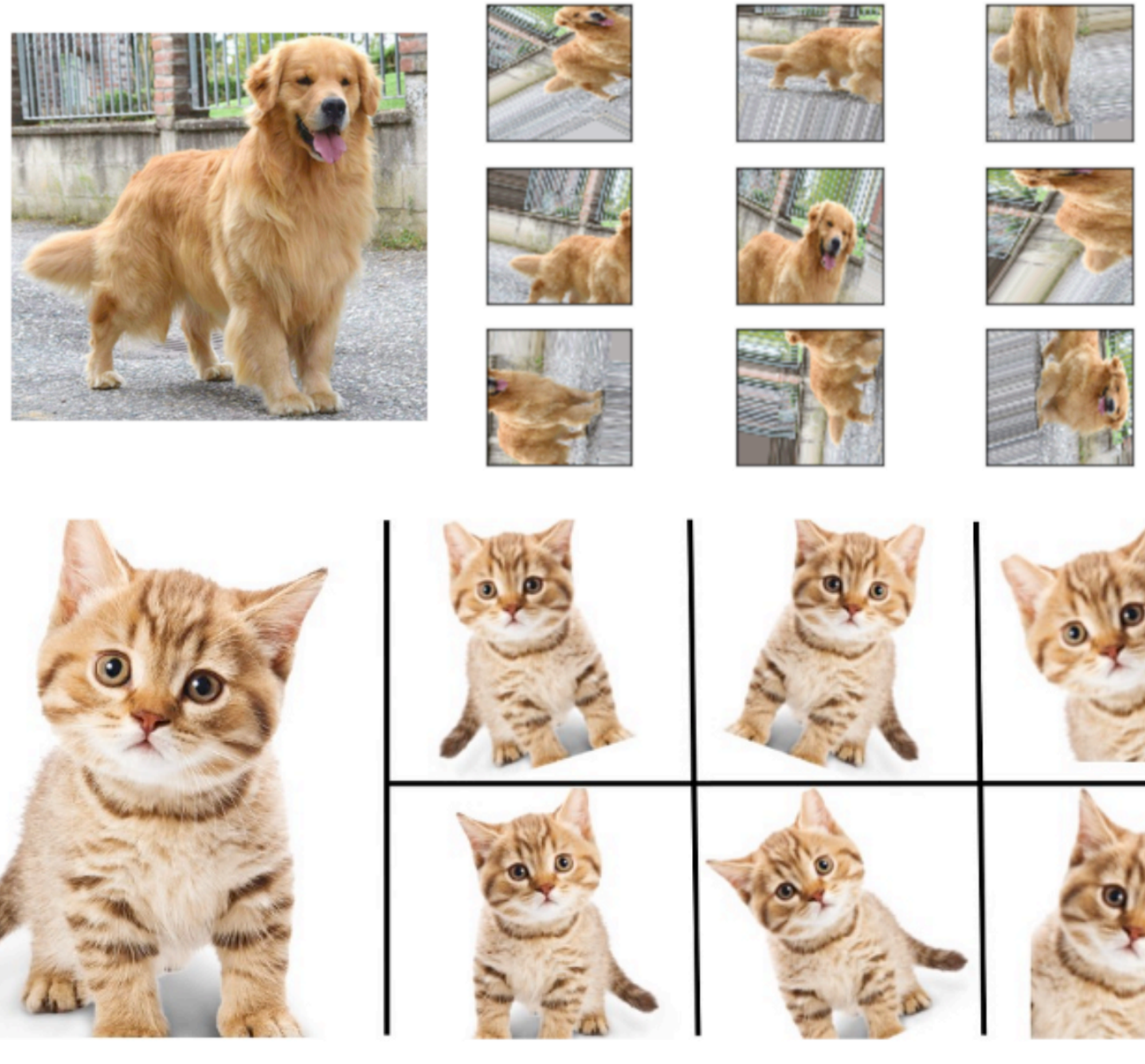
Data augmentation

- Other forms of transformation
  - Random cropping
  - Whitening
  - Center the input mean to zero
  - Random Affine transformation
  - Random Brightness shifting
  - Random width/height shifting
  - Custom, etc.

Read more, section 7.4 of
this book

Data augmentation

- Augmenting your dataset can increase the performances of your model



Pretrained Layers

Pretrained Layers

- The use of DNN model that weights/layers are (already) optimized for a particular task is a common practice.
  Two scenarios:
    - **Unfreezed**: The parameters of the pertained layers can be updated during the training. Avoid training from scratch. Useful to speedup the training instead of starting from scratch.
    - **Freezed**: parameters of the pre-trained layers are not updatable, i.e., the model pre-trained on a given task, is used as feature extractor for another task. This is called **Transfer Learning**

Pretrained layers

Transfer Learning

- Intuition

Task 1

Knowledge transfer

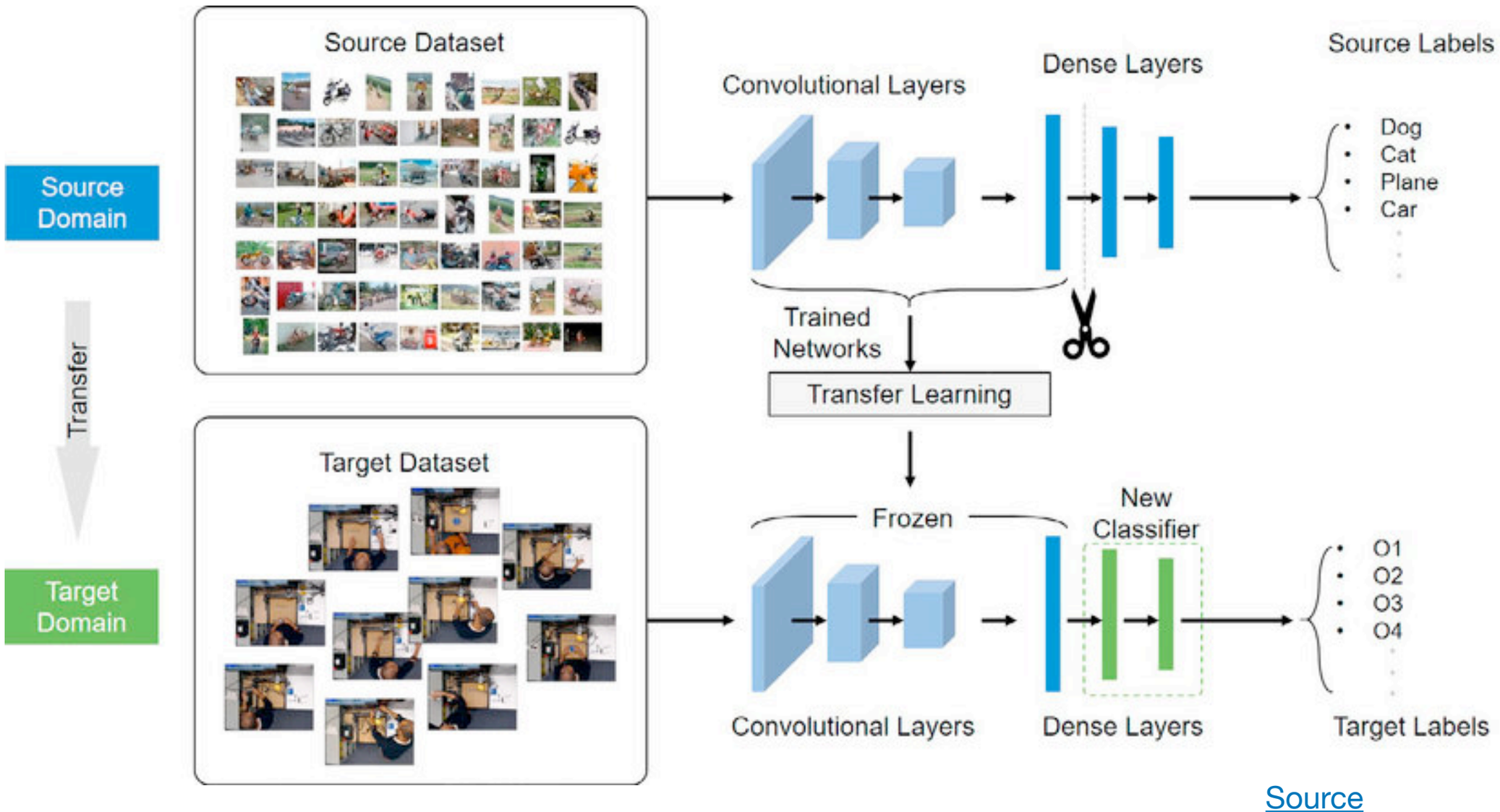
Task 2

Assumption: Many of the factors that explain the variations in Task1 are relevant to the variations that need to be captured for the learning Task 2

Source Images: VectorsStocks

Pretrained Layers

- Transfer learning generally leverages the learning that was done on huge datasets and on strong resources that's not easily accessible.

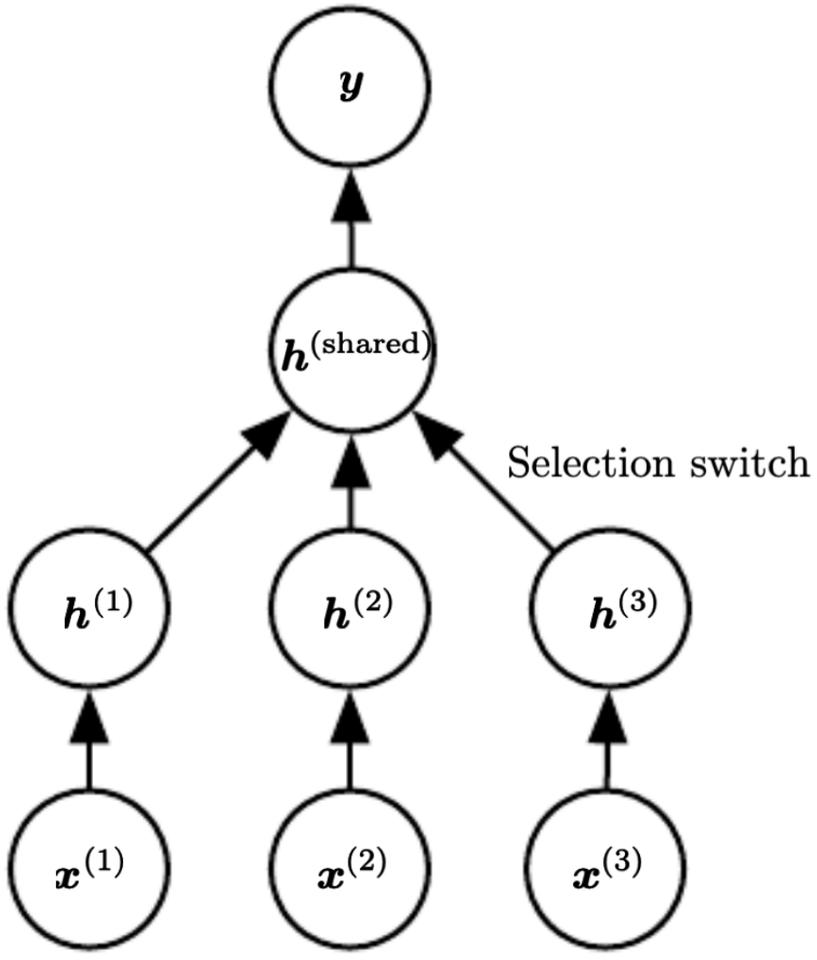


Source

Pretrained Layers

Sharing the later layers

- Sometimes it isn't the input semantic/nature is shared between tasks, but the output itself.



Read more, chapter 15 of the book (Must read)

Summary

- Regularization techniques for DNNs
  - Dropout, Batch Normalization, Early Stopping, Data augmentation
- Optimizers
  - Adam, RMSProps, SGD with momentum
- Data augmentation
  - Enlarge the dataset by applying some transformations on the data
- Pretrained Layers