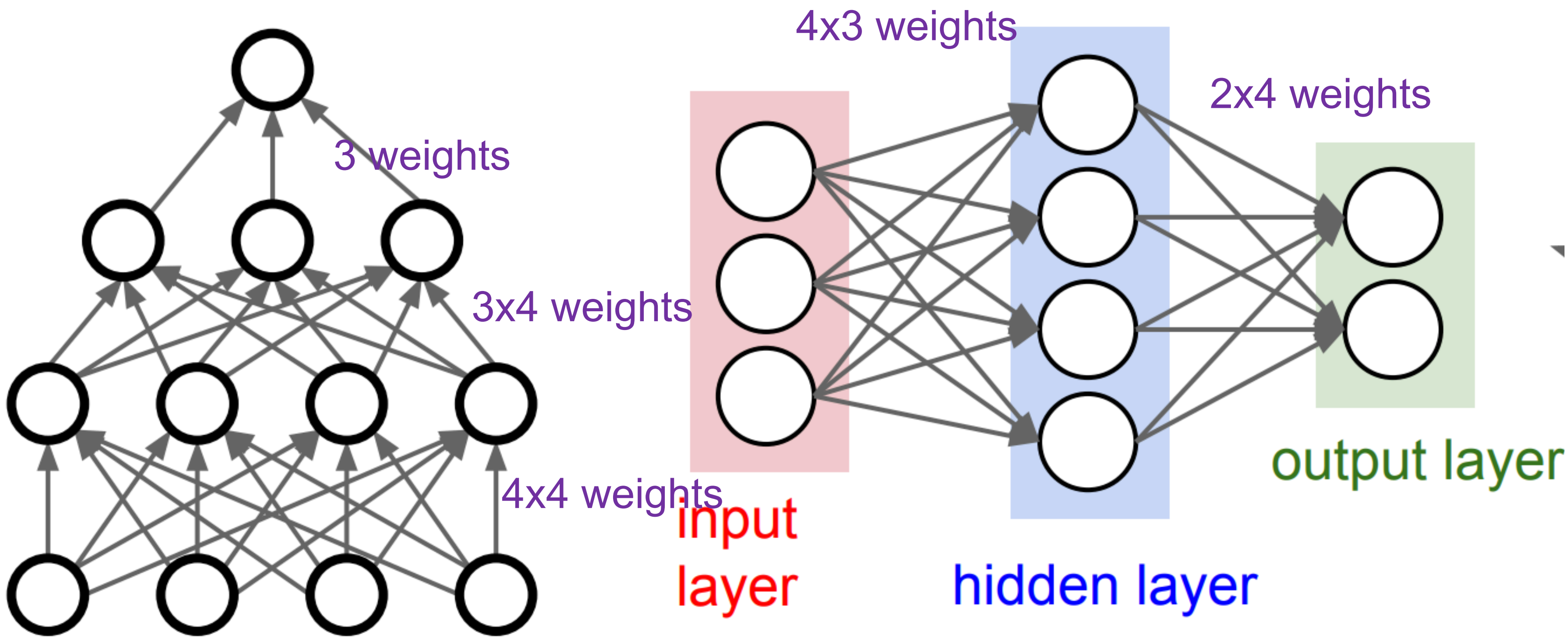
CMSC 510 – L19
Regularization Methods for Machine Learning

Part 19a: Dropout

Instructor:
Dr. Tom Arodz

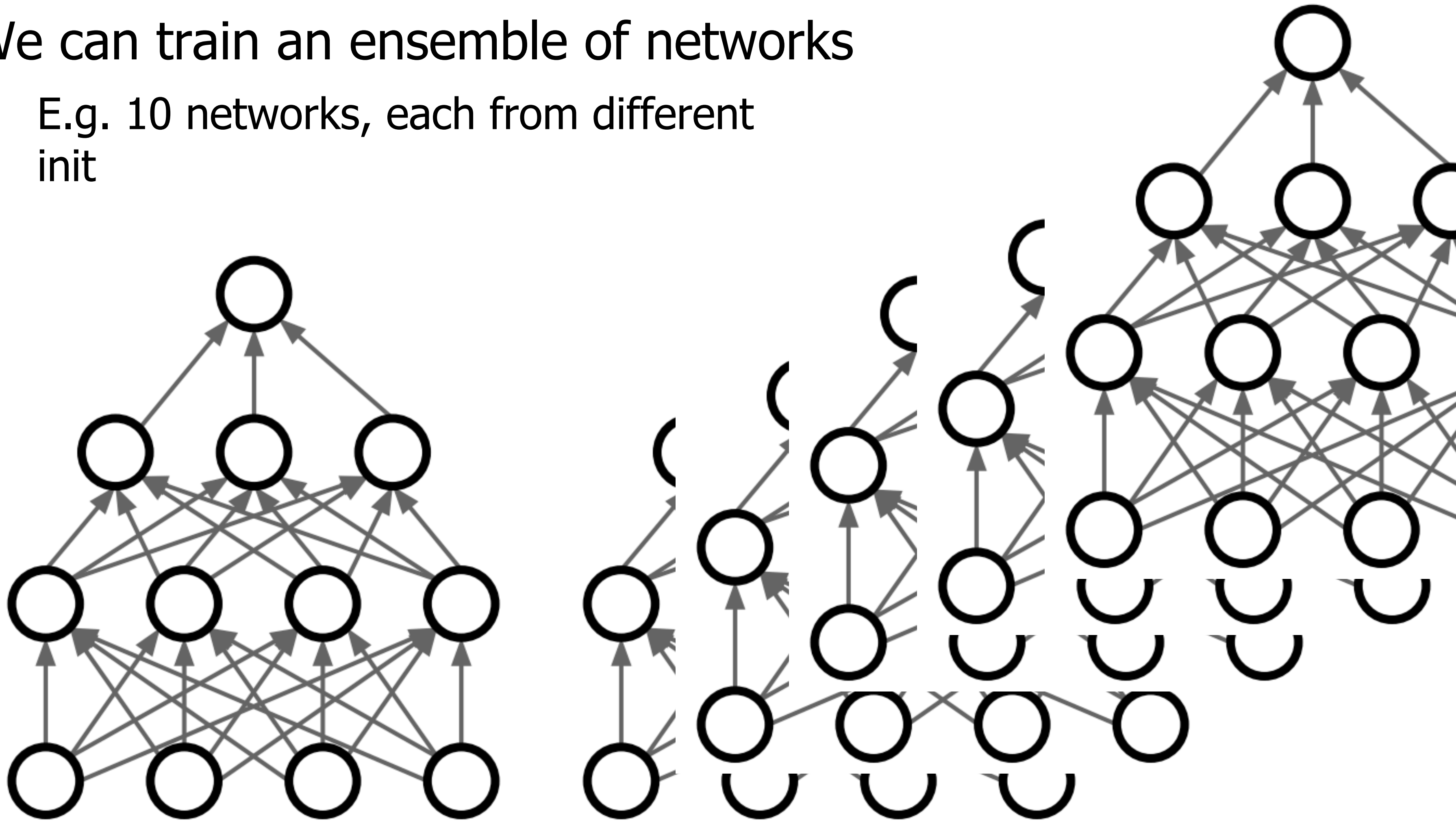
Regularization in deep nets

- A feedforward network has many many weights



Regularization in deep nets

- Each different initialization from random weights will lead to a different network
- We can train an ensemble of networks
  - E.g. 10 networks, each from different init

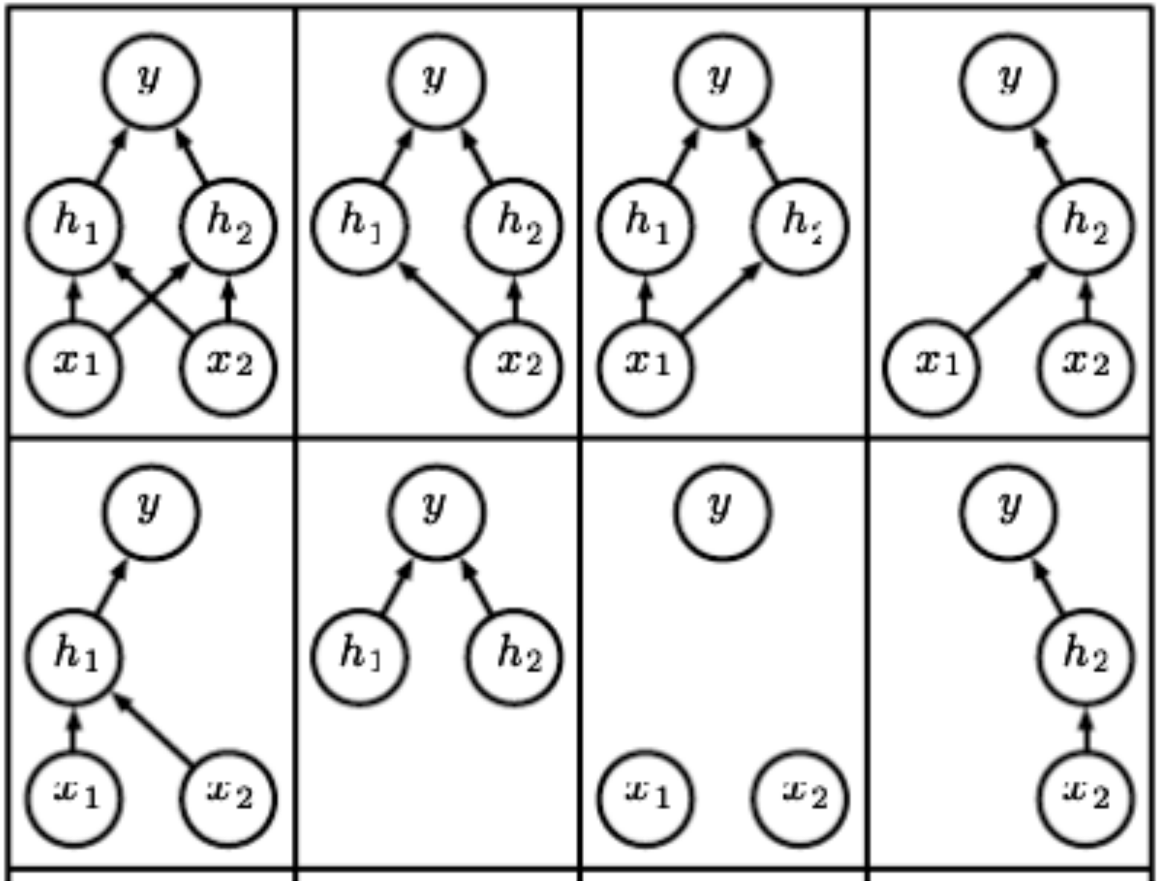


Regularization in deep nets

- We can also train an ensemble of networks with slightly different architectures

- A large ensemble:

=> a lot of weights!



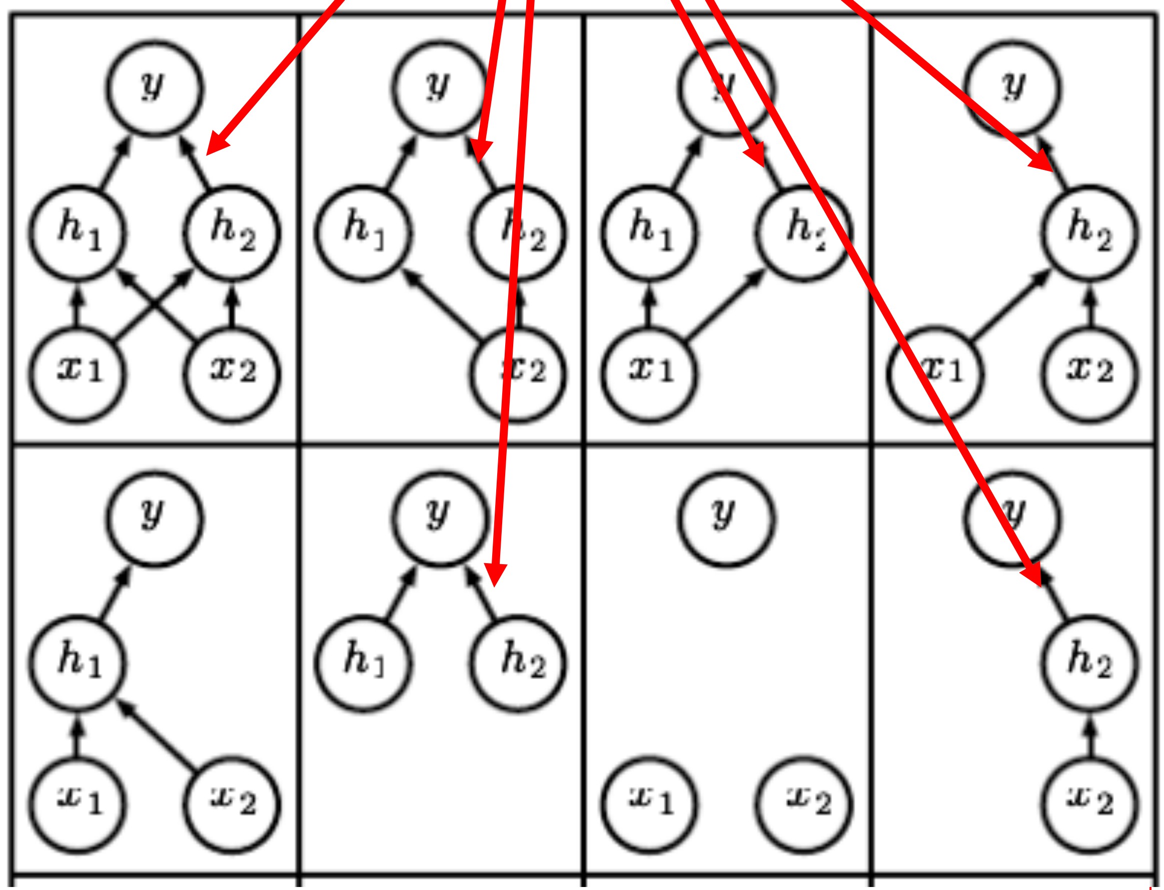
Regularization in deep nets

- We can also train an ensemble of networks with slightly different architectures
- A large ensemble:

=> a lot of weights!

- How to reduce number of weights?
  - Weight sharing

- Same “shared” weight



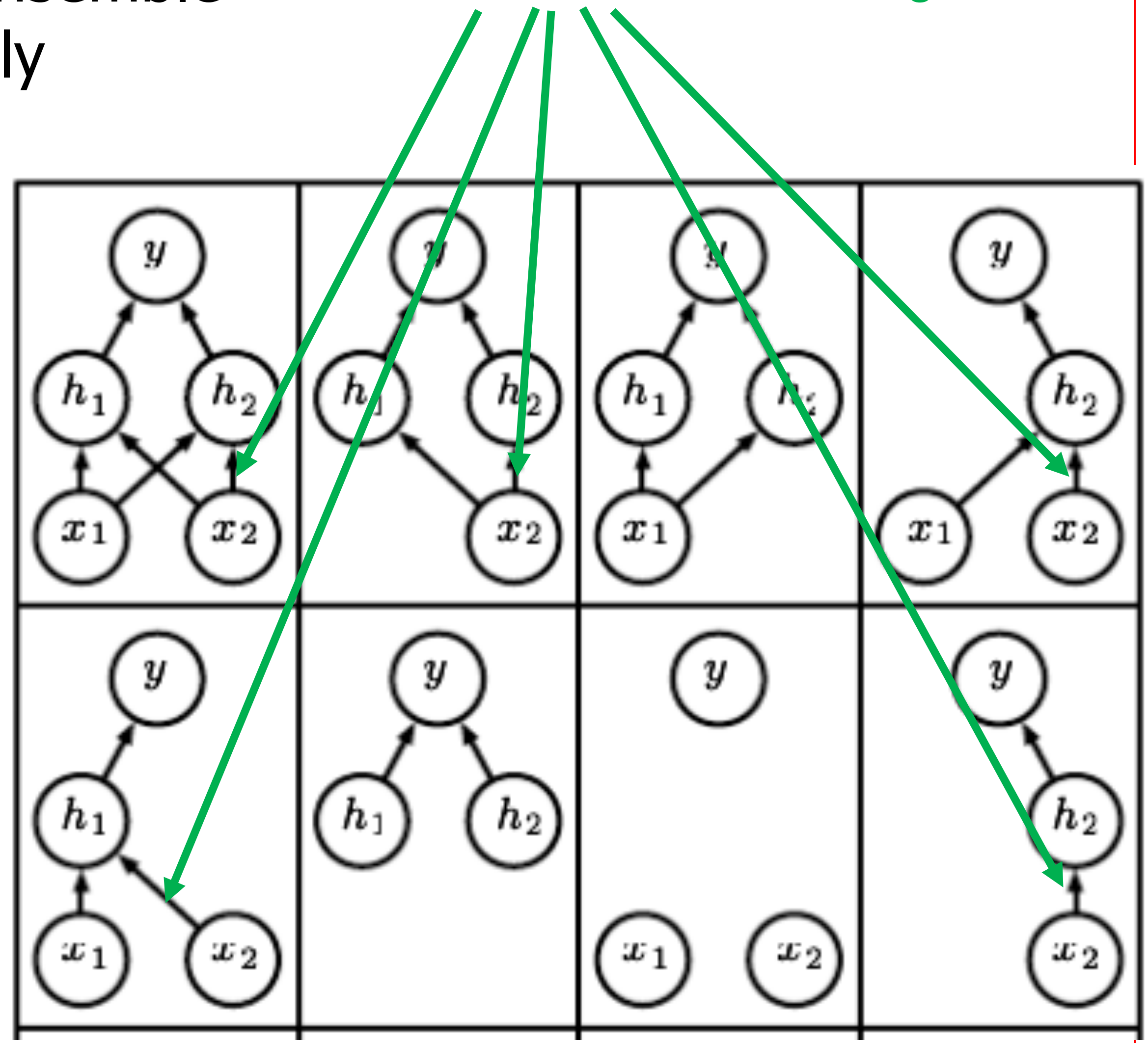
Regularization in deep nets

- We can also train an ensemble of networks with slightly different architectures
- A large ensemble:

=> a lot of weights!

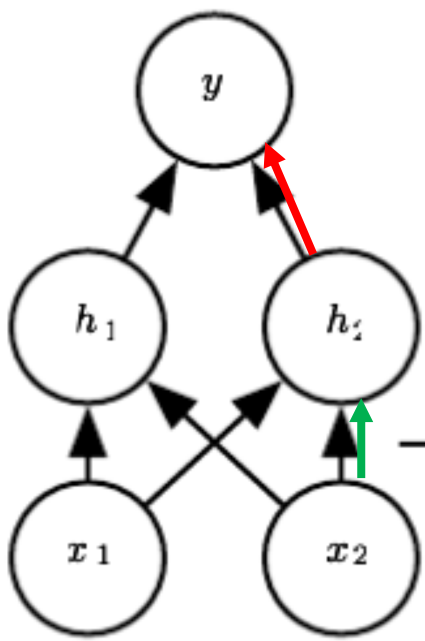
- How to reduce number of weights?
  - Weight sharing

- Same "shared" weight

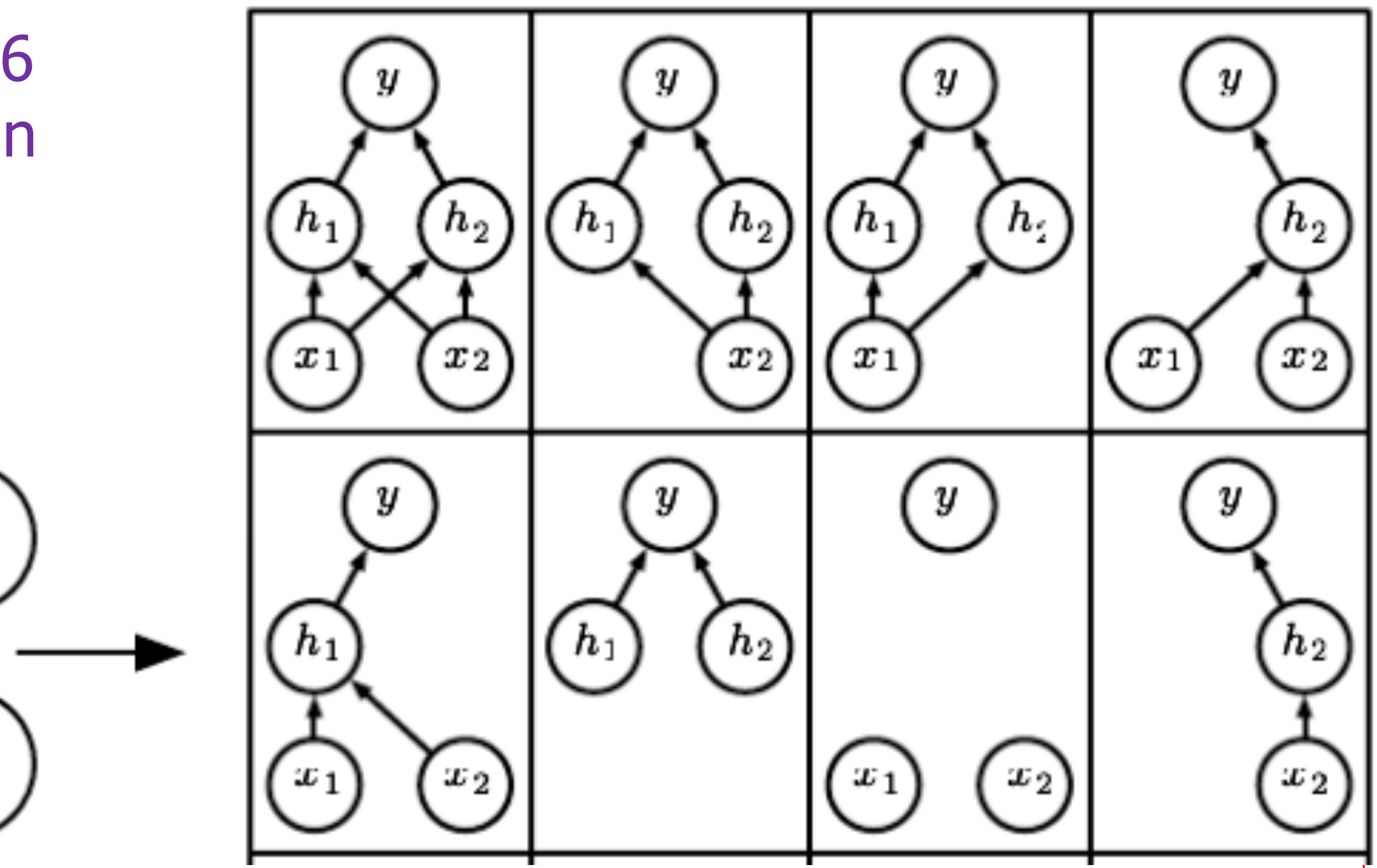


Regularization in deep nets

- How to reduce number of weights?
  - Weight sharing
- We only have 6 weights to train

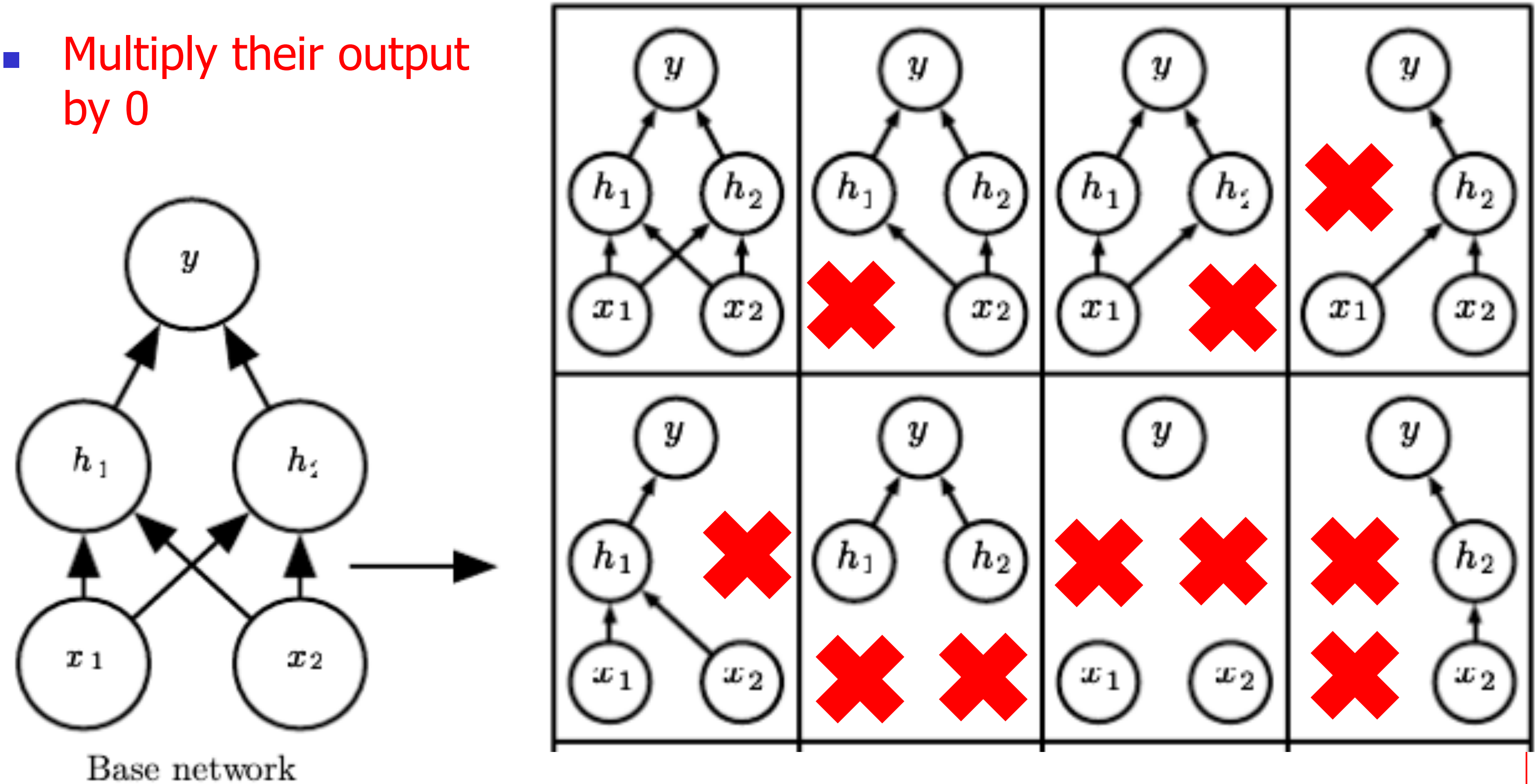


Base network



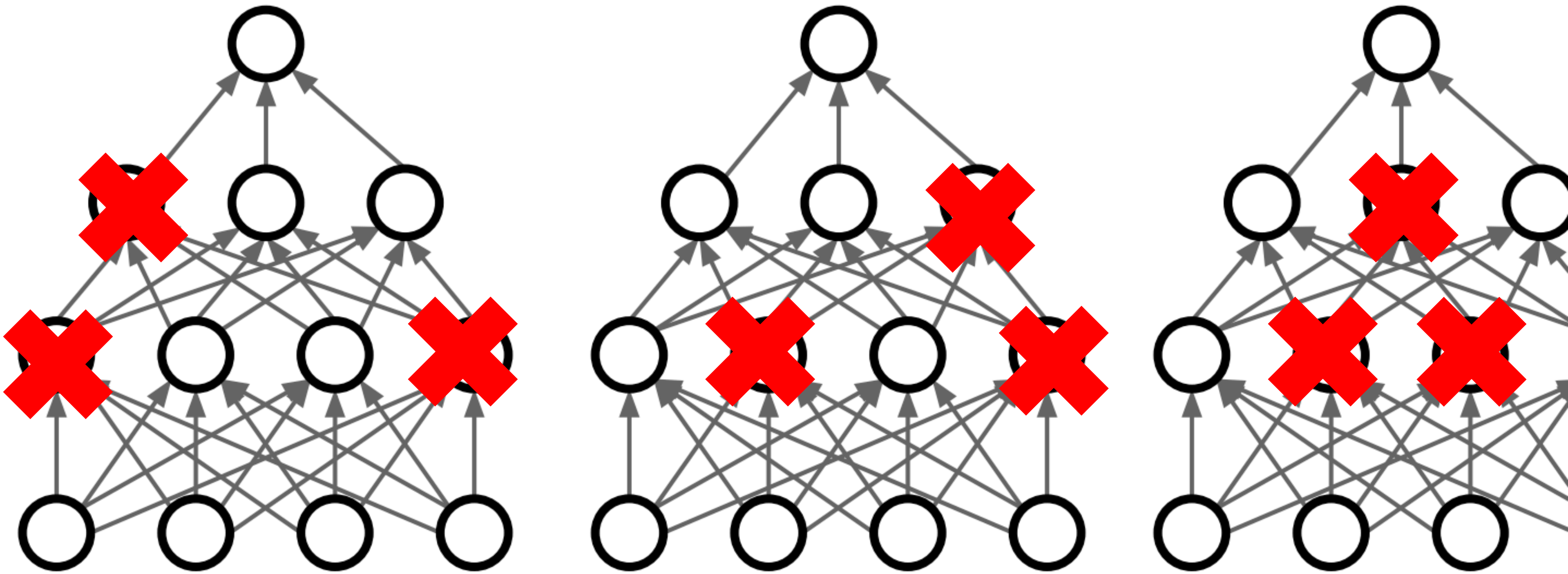
Dropout technique

- Dropout: how to train all these ensembles?
- Keep the base network with its weights, just randomly “drop” neurons
  - Multiply their output by 0



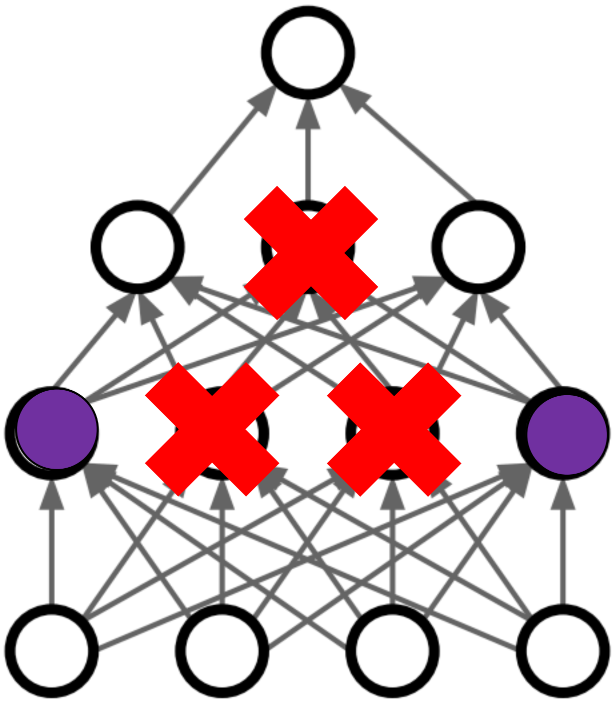
Dropout technique

- Weight sharing in an ensemble
  - Dropout – train multiple networks, each with some neurons missing
  - The edges/weights are kept the same across networks



Dropout technique

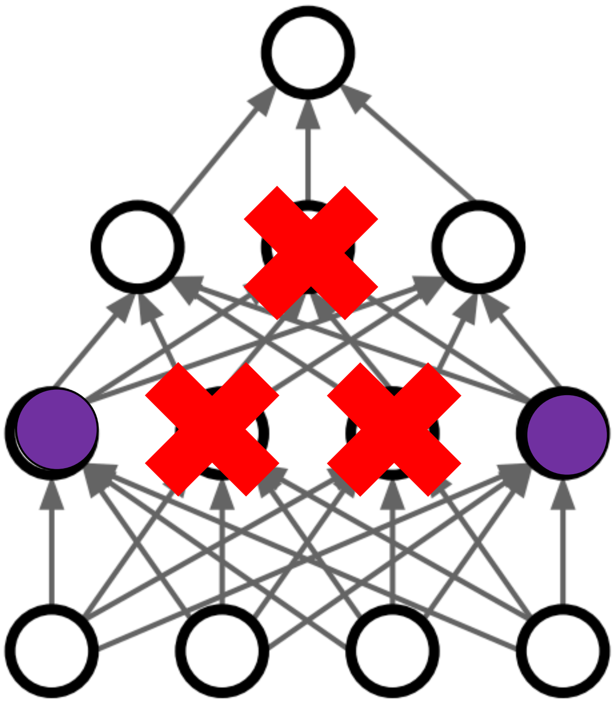
- Dropout – during testing:
  - Using the whole network "as is" is not optimal
  - During training we e.g. in layer 3 have input from e.g. two neurons
  - If during testing we use all four neurons, we'll have a higher input to layer 3 neurons
- Solution:
  - Reduce the weights after training
  - E.g. if you drop out 50% of neurons
    - Then reduce weights by 2



Dropout technique

- E.g. if you drop out 50% of neurons
  - Then reduce weights by 2
- It's an approximation:
  - True solution is to multiply outputs p(y|x) by p(mu), probability of the drop pattern

$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$$



  - Difficult, needs to actually run all networks
  - Alternative: use geometric mean

$$\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x}) = \sqrt[2^d]{\prod_{\boldsymbol{\mu}} p(y \mid \boldsymbol{x}, \boldsymbol{\mu})}$$

Dropout technique

- Alternative: use geometric mean
  - Weights v, binary drop vector d

$$\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) = \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d})}$$

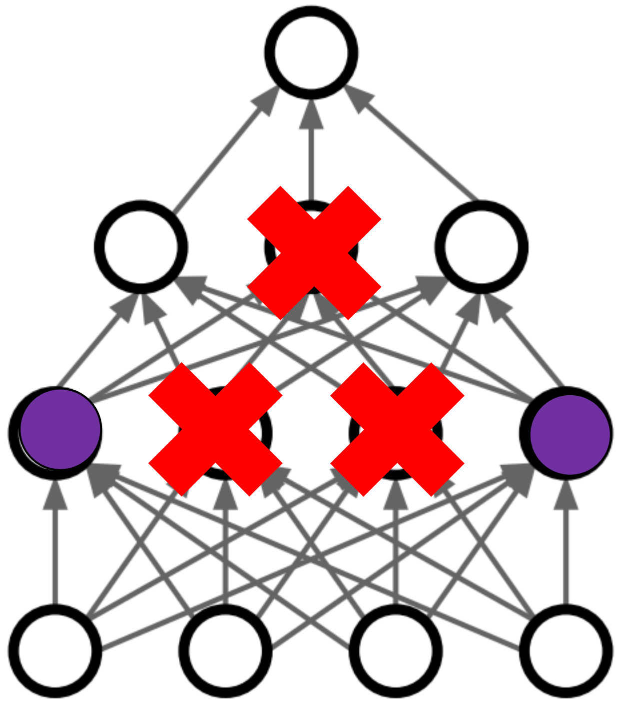
  - For softmax + just linear layer: $P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d}) = \text{softmax}\left(\boldsymbol{W}^\top (\boldsymbol{d} \odot \mathbf{v}) + \boldsymbol{b}\right)_y$

$$\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) = \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d})}$$

$$= \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \text{softmax}\left(\boldsymbol{W}^\top (\boldsymbol{d} \odot \mathbf{v}) + \boldsymbol{b}\right)_y}$$

$$= \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \frac{\exp\left(\boldsymbol{W}_{y,:}^\top (\boldsymbol{d} \odot \mathbf{v}) + b_y\right)}{\sum_{y'} \exp\left(\boldsymbol{W}_{y',:}^\top (\boldsymbol{d} \odot \mathbf{v}) + b_{y'}\right)}}$$

$$= \frac{\sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \exp\left(\boldsymbol{W}_{y,:}^\top (\boldsymbol{d} \odot \mathbf{v}) + b_y\right)}}{\sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \sum_{y'} \exp\left(\boldsymbol{W}_{y',:}^\top (\boldsymbol{d} \odot \mathbf{v}) + b_{y'}\right)}}$$



Dropout technique

- Alternative: use geometric mean
  - Weights v, binary drop vector d

$$\tilde{P}_{\text{ensemble}}(\mathrm{y}=y \mid \mathbf{v}) = \sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n} P(\mathrm{y}=y \mid \mathbf{v};\boldsymbol{d})}$$

  - For softmax + just linear layer: $P(\mathrm{y}=y \mid \mathbf{v};\boldsymbol{d}) = \text{softmax}\left(\boldsymbol{W}^\top(\boldsymbol{d}\odot\mathbf{v})+\boldsymbol{b}\right)_y$

$$\tilde{P}_{\text{ensemble}}(\mathrm{y}=y \mid \mathbf{v}) \propto \sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n} \exp\left(\boldsymbol{W}_{y,:}^\top(\boldsymbol{d}\odot\mathbf{v})+b_y\right)}$$

$$= \exp\left(\frac{1}{2^n}\sum_{\boldsymbol{d}\in\{0,1\}^n} \boldsymbol{W}_{y,:}^\top(\boldsymbol{d}\odot\mathbf{v})+b_y\right)$$

$$= \exp\left(\boxed{\frac{1}{2}}\boldsymbol{W}_{y,:}^\top\mathbf{v}+b_y\right).$$

  - For general net, cutting weights by half is just an approximation

Dropout technique summary

- Dropout – during training:
  - Define a single network
  - In each iteration (mini-batch), select random 50% of neurons and multiply their output by 0
- Dropout – during testing:
  - Use the single trained network
  - Just divide all weights by 2