Foundations of DL

Deep Learning

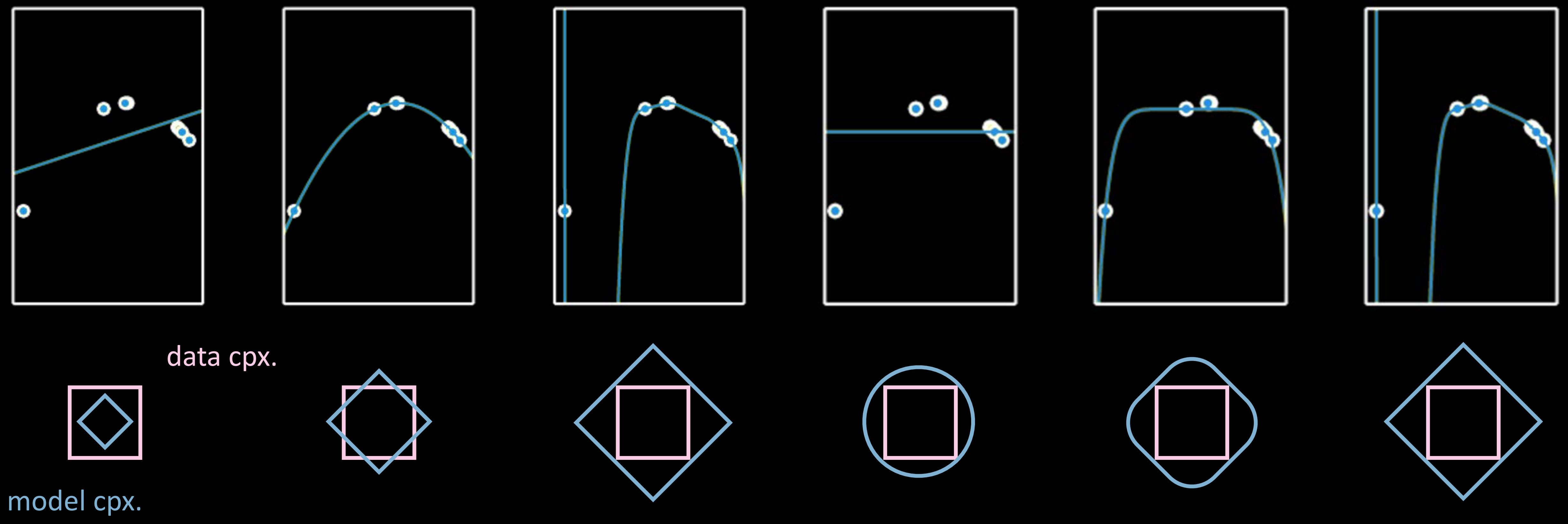
ALF

Alfredo Canziani, Ritchie Ng
@alfcnz, @RitchieNg

Overfitting and regularisation

Connection between them

Model selection and regularisation



underfitting | right-fitting | overfitting | strong reg. | medium reg. | weak reg.

Regularisation – definitions

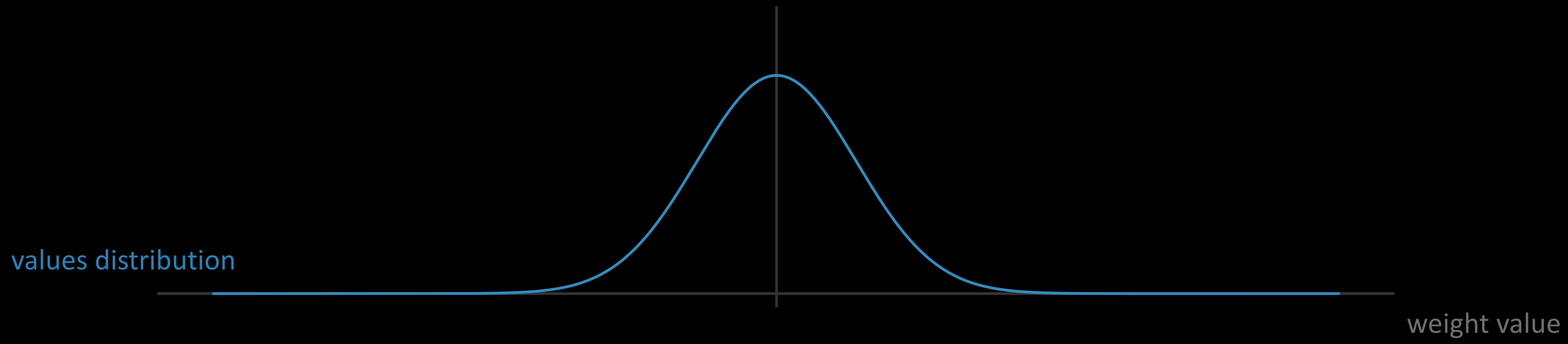
- Regularisation adds prior knowledge to a model;
  a prior distribution is specified for the parameters
- Restriction of set of possible learnable functions
- Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error -- Ian Goodfellow

Regularising techniques

A few examples

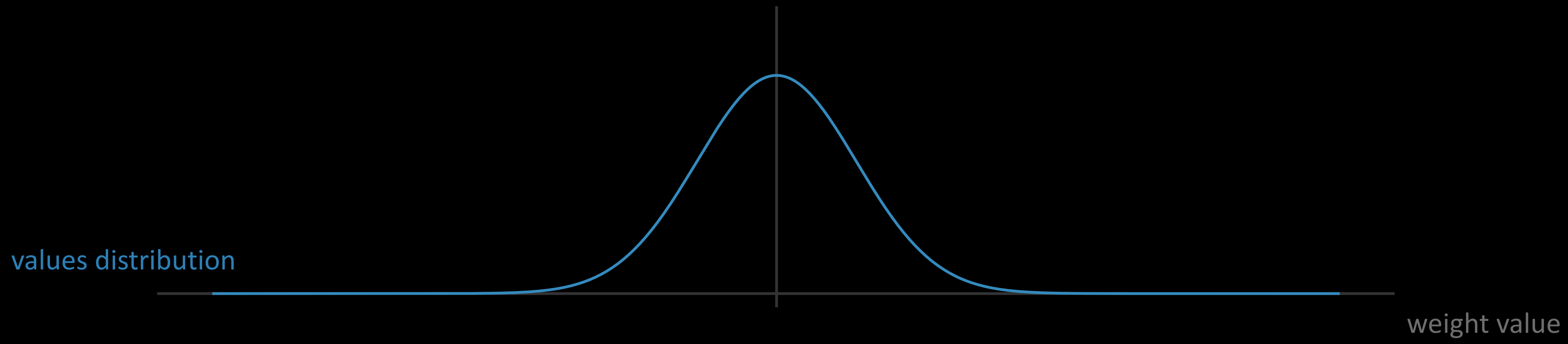
Xavier (initialising techniques)

- Xavier
  - `torch.nn.init.xavier_normal_(tensor, gain=1)`
  - Docs: pytorch.org/docs/master/nn.html#torch.nn.init.xavier_normal_
  - Author
    - Xavier Glorot



Weight-decay

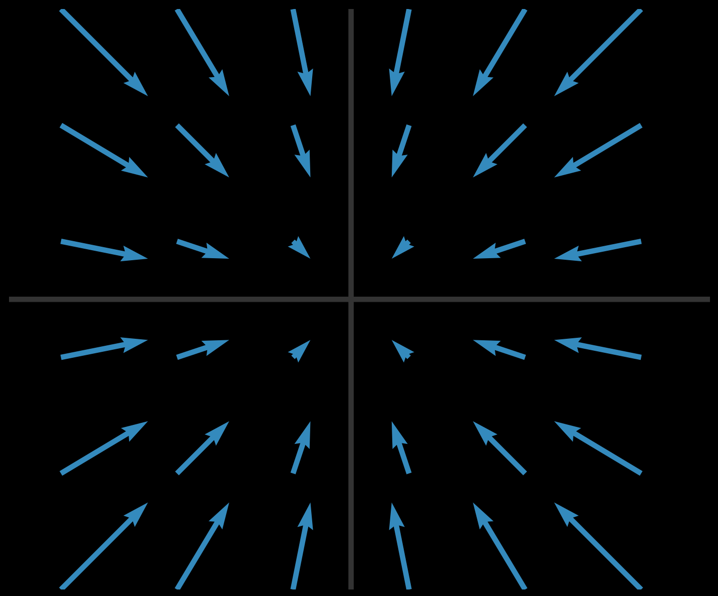
- Weight-decay
  - Docs: pytorch.org/docs/master/optim
  - Alternative names
    - L2
    - Ridge
    - Gaussian prior

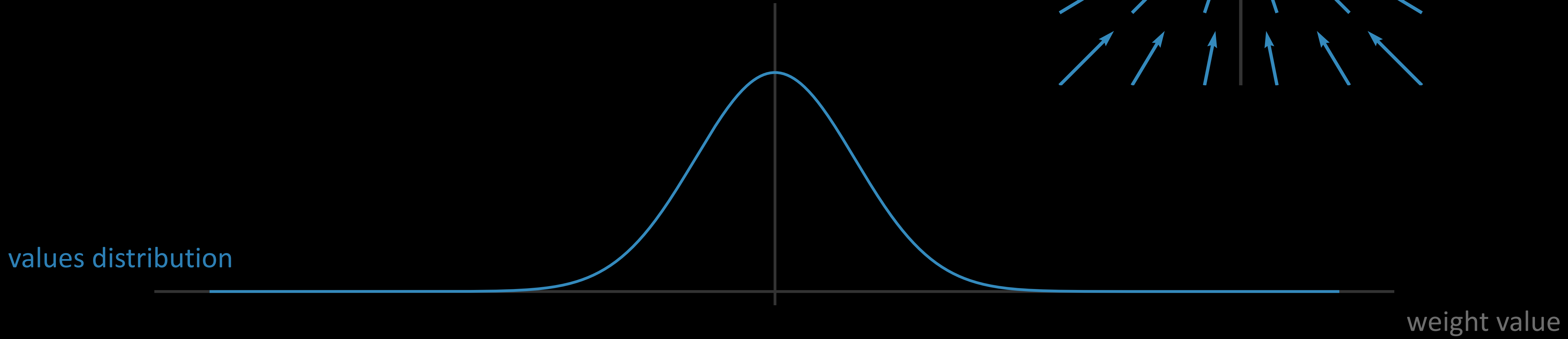


Weight-decay

$$J_{\text{train}}(\boldsymbol{\theta}) = J_{\text{train}}^{\text{old}}(\boldsymbol{\theta}) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

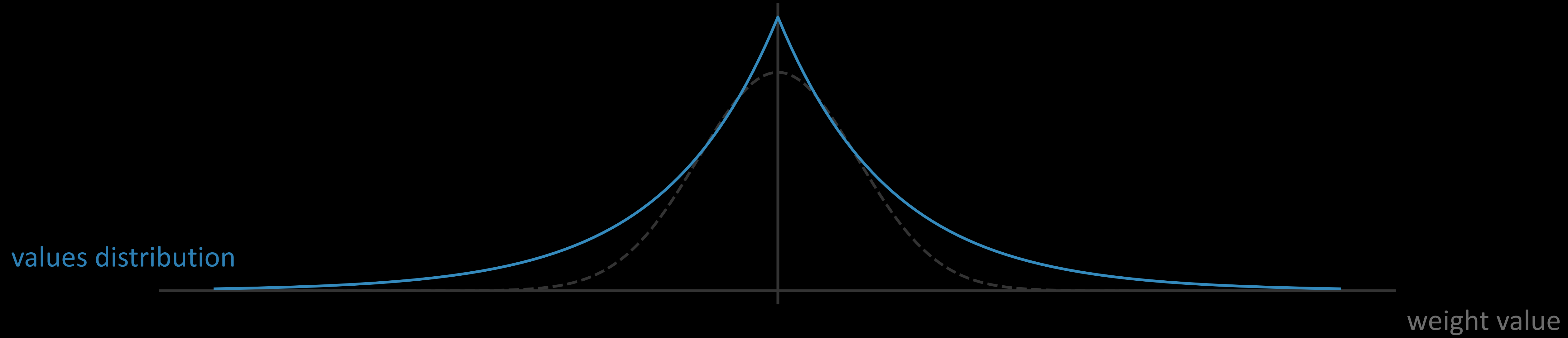
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J_{\text{train}}^{\text{old}}(\boldsymbol{\theta}) - \eta\lambda\boldsymbol{\theta}$$





L1

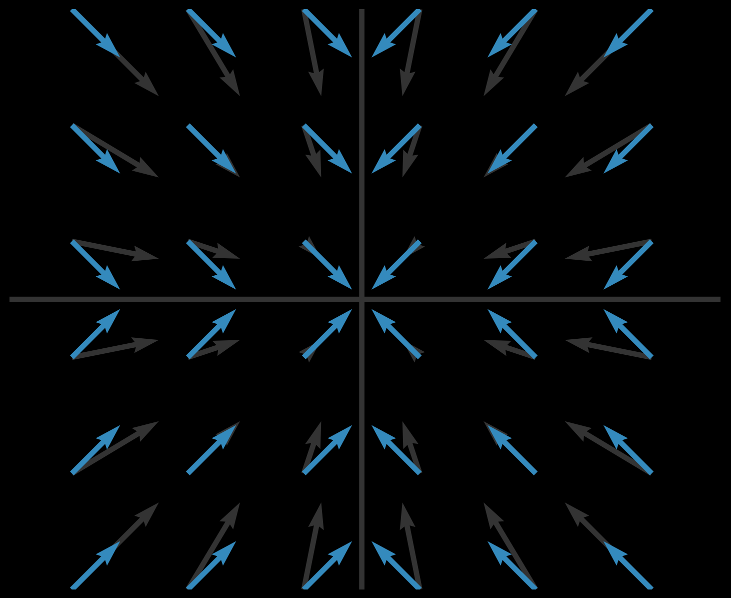
- L1
  - Docs: pytorch.org/docs/master/optim
  - Alternative names
    - LASSO: Least Absolute Shrinkage Selector Operator
    - Laplacian prior
    - Sparsity prior

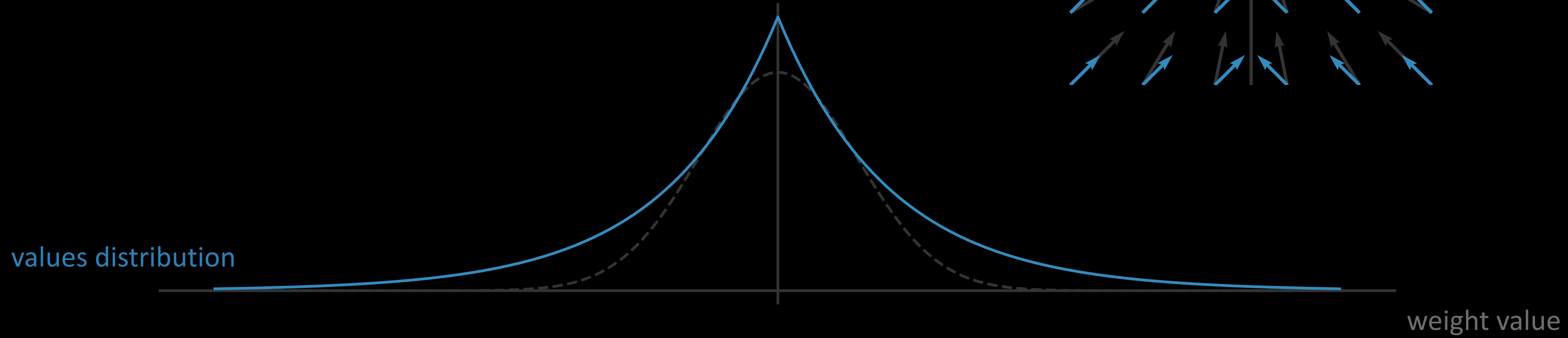


L1

$$J_{\text{train}}(\boldsymbol{\theta}) = J_{\text{train}}^{\text{old}}(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1$$

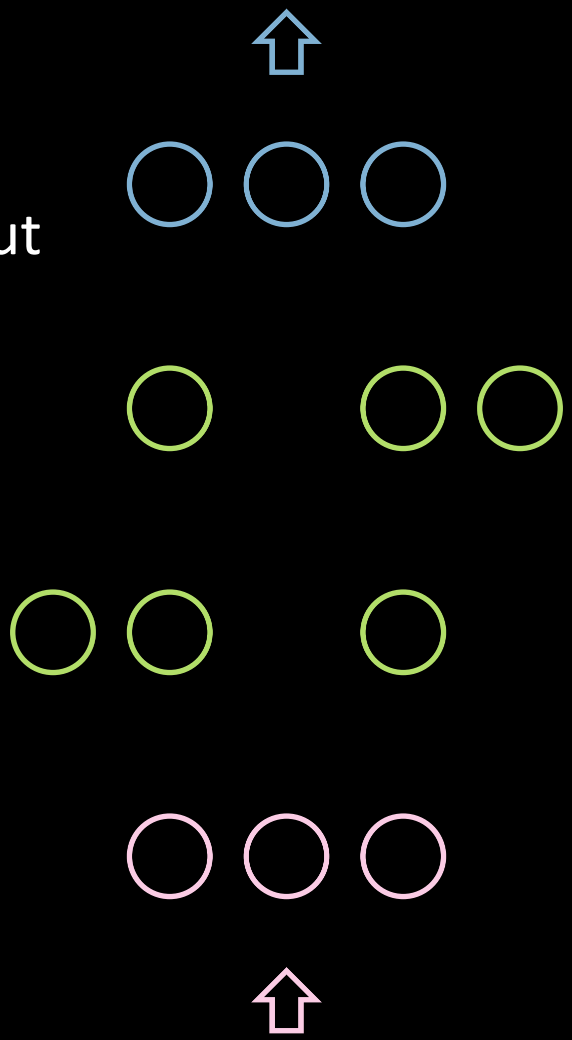
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J_{\text{train}}^{\text{old}}(\boldsymbol{\theta}) - \eta \lambda \, \text{sign}(\boldsymbol{\theta})$$





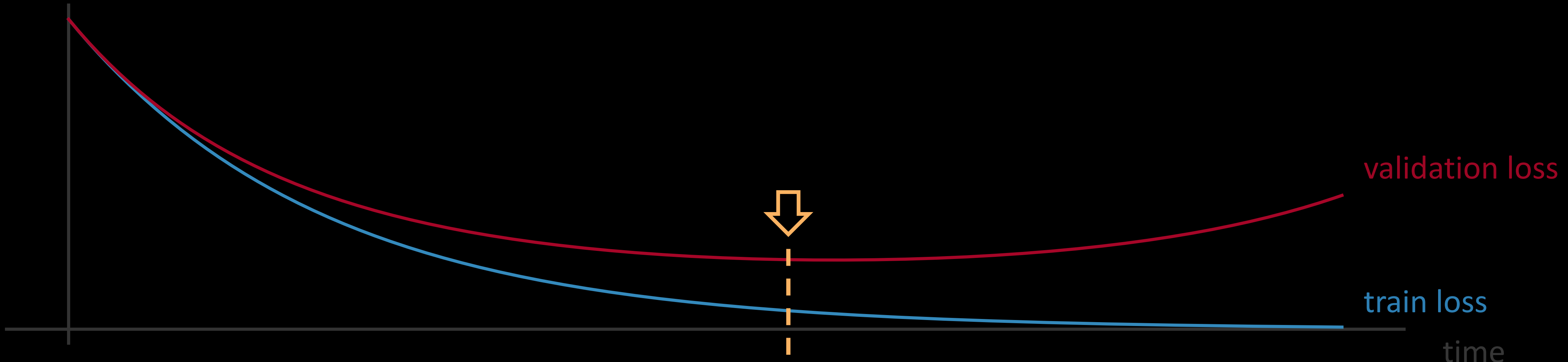
Dropout

- Dropout
  - torch.nn.Dropout(rate=0.5)
  - Docs: pytorch.org/docs/master/nn.html#torch.nn.Dropout
  - Variants
    - torch.nn.Dropout2d(rate=0.5)
    - torch.nn.Dropout3d(rate=0.5)
    - torch.nn.AlphaDropout(rate=0.5)



Early-stopping

- Early-stopping
  - `if acc > best_acc: torch.save(model, 'model.pth')`

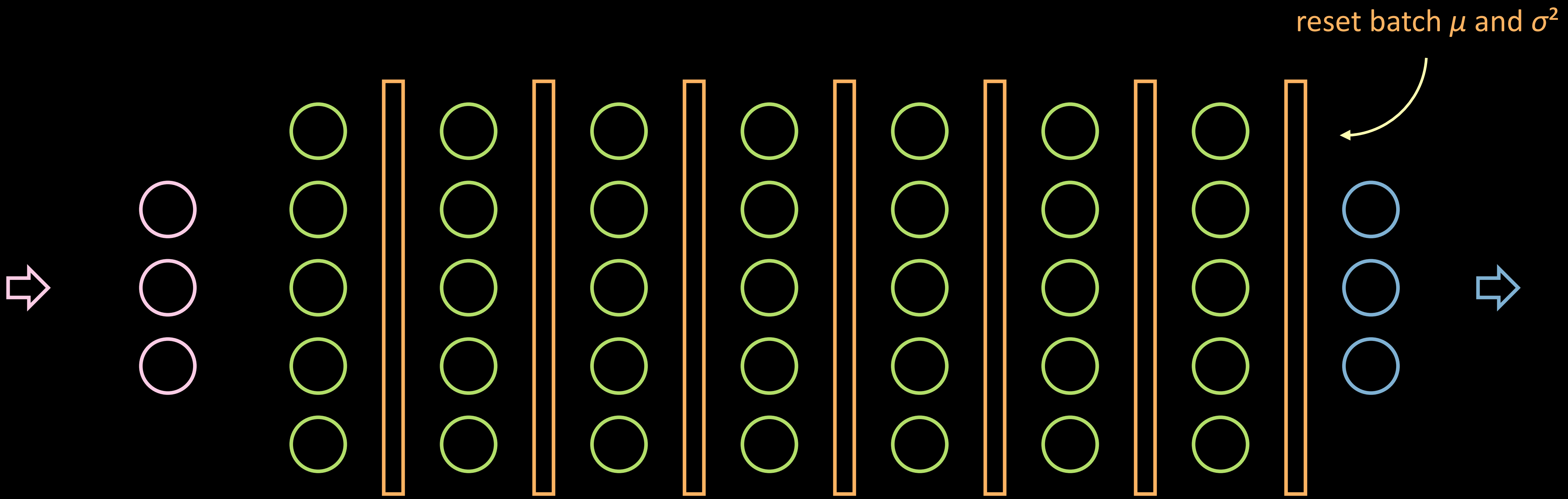


Fighting overfitting

Techniques that ends up regularising our parameters

Batch-norm (regularisation by-product)

- Batch-normalisation
  - `torch.nn.BatchNorm1d(num_features)`
  - Docs: pytorch.org/docs/master/nn.html #batchnorm1d



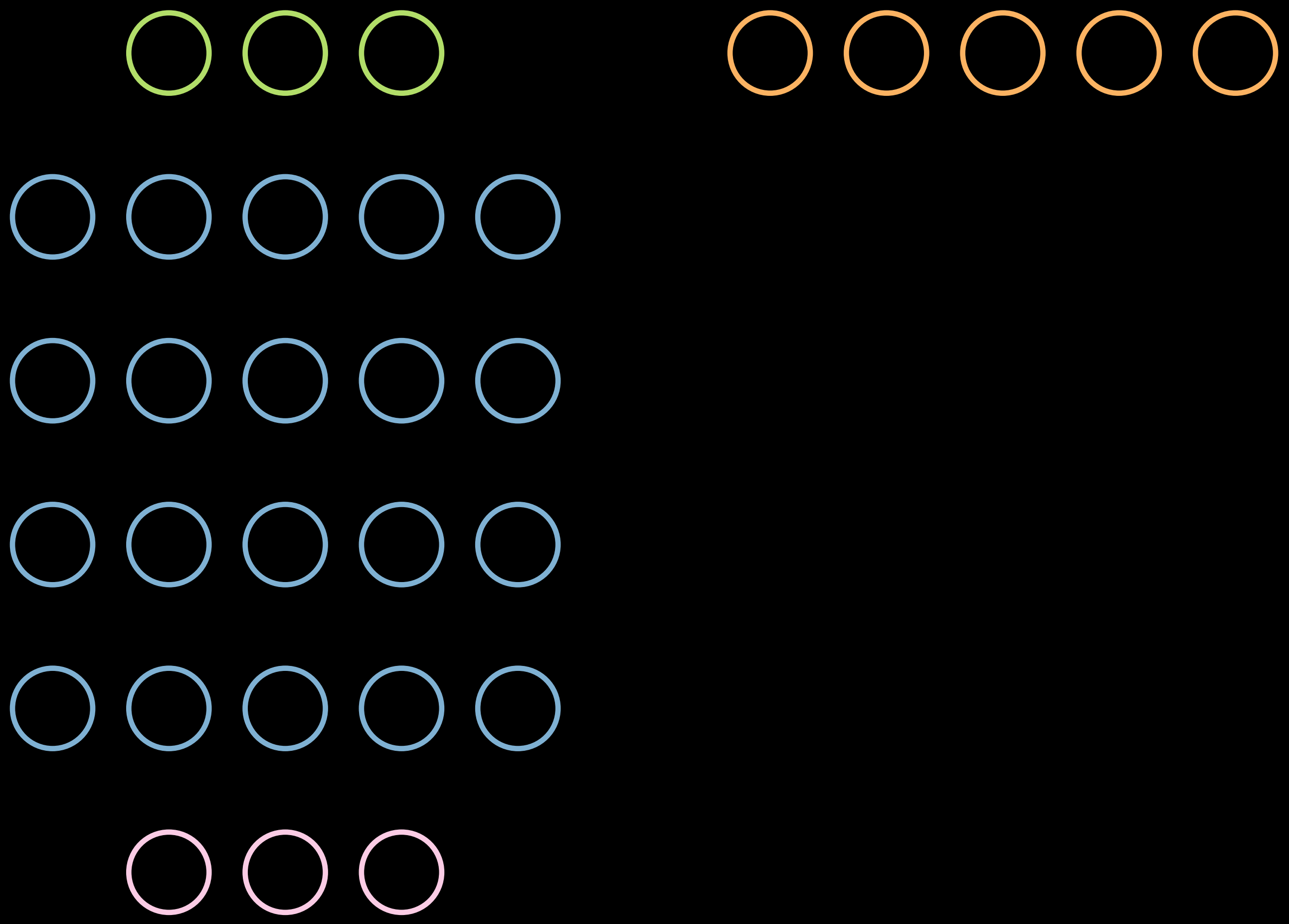
More-data

- More-data
    - $$$

Data-augmentation

- Data-augmentation
  - `torchvision.transforms.Compose(transforms)`
  - Docs: pytorch.org/docs/stable/torchvision/transforms.html
  - Tranformations
    - `torchvision.transforms.CenterCrop(size)`
    - `torchvision.transforms.ColorJitter(brightness, contrast, saturation, hue)`
    - `torchvision.transforms.FiveCrop(size)`
    - `torchvision.transforms.LinearTransformation(transformation_matrix)`
    - `torchvision.transforms.RandomAffine(degrees, translate, scale, shear)`
    - `torchvision.transforms.RandomCrop(size, padding, pad_if_needed, fill)`
    - `torchvision.transforms.RandomRotation(degrees)`
    - `torchvision.transforms.RandomHorizontalFlip(p=0.5)`

Transfer learning (TL) & fine tuning (FT)



- Few data ~ train ⇒ TL
- Lots data ~ train ⇒ FT
- Few data ! train ⇒ early TL
- Lots data ! train ⇒ T

- Use diversified learning rates

remove a few more layers from the top