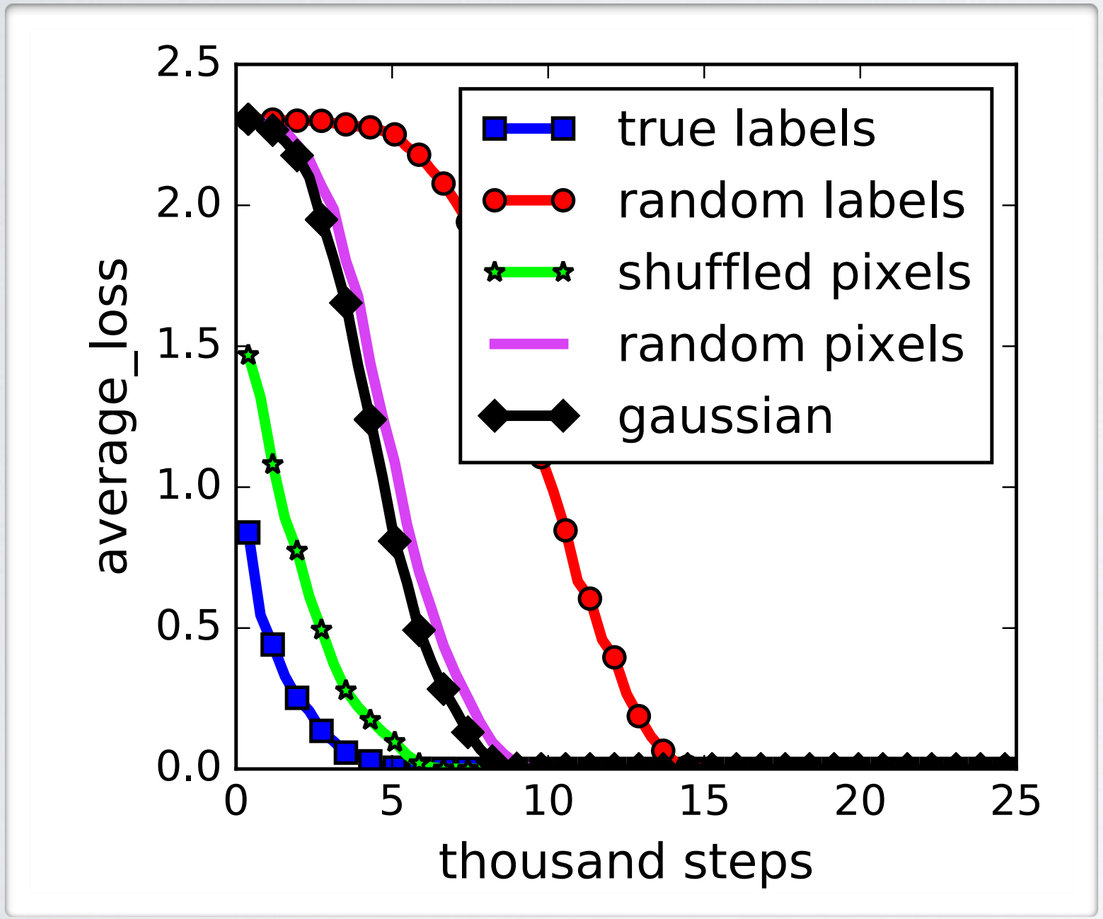
Review of material from Lecture 02: Neural Network Training

NEURAL NETS CAN EASILY MEMORIZE

Understanding Deep Learning Requires Rethinking Generalization
Zhang, Bengio, Hardt, Recth, Vinyals, ICLR 2017



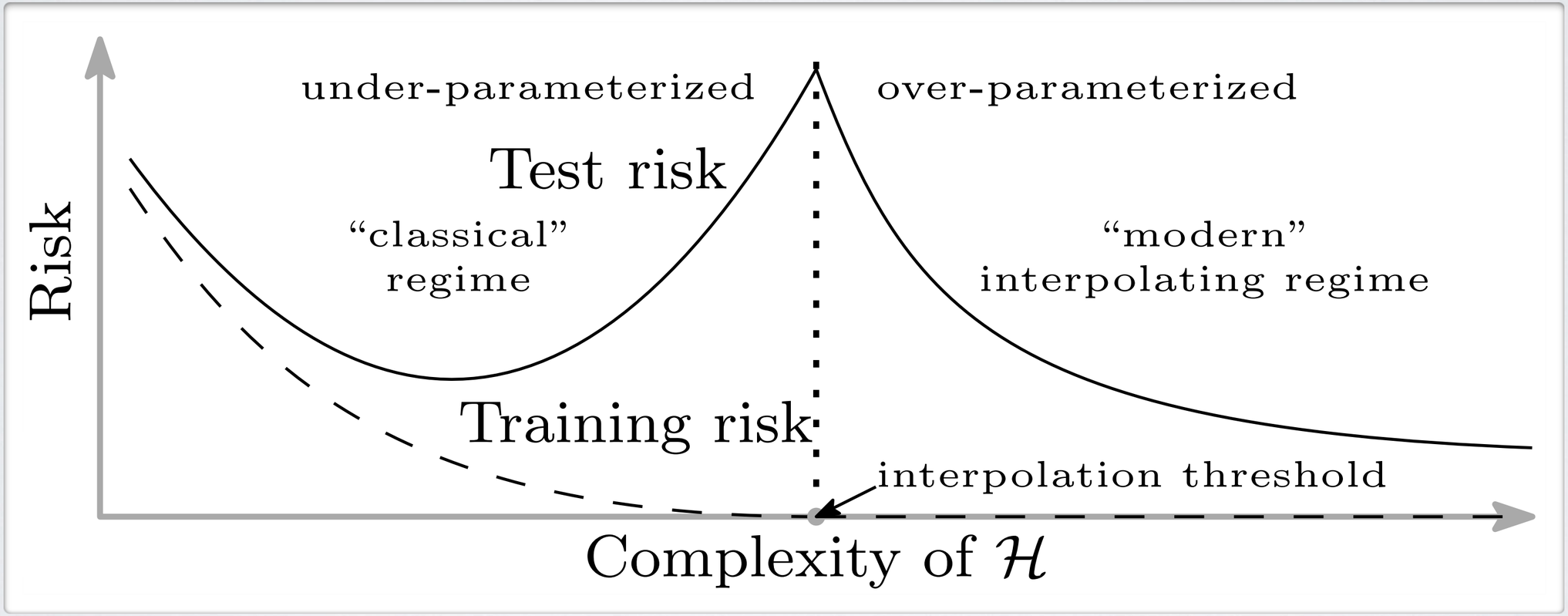
Inception model on the CIFAR10 dataset

- **True labels**: the original dataset without modification.
- **Random labels**: all the labels are replaced with random ones.
- **Shuffled pixels**: a random permutation of the pixels is chosen and then the same permutation is applied to all the images in both training and test set.
- **Random pixels**: a different random permutation is applied to each image independently.
- **Gaussian**: A Gaussian distribution (with matching mean and variance to the original image dataset) is used to generate random pixels for each image.

THEY UNDERFIT/OVERFIT STRANGELY

Topics: bias/variance trade-off, interpolation threshold

- *Reconciling modern machine learning and the bias-variance trade-off*
  Belkin et al. arXiv 2018



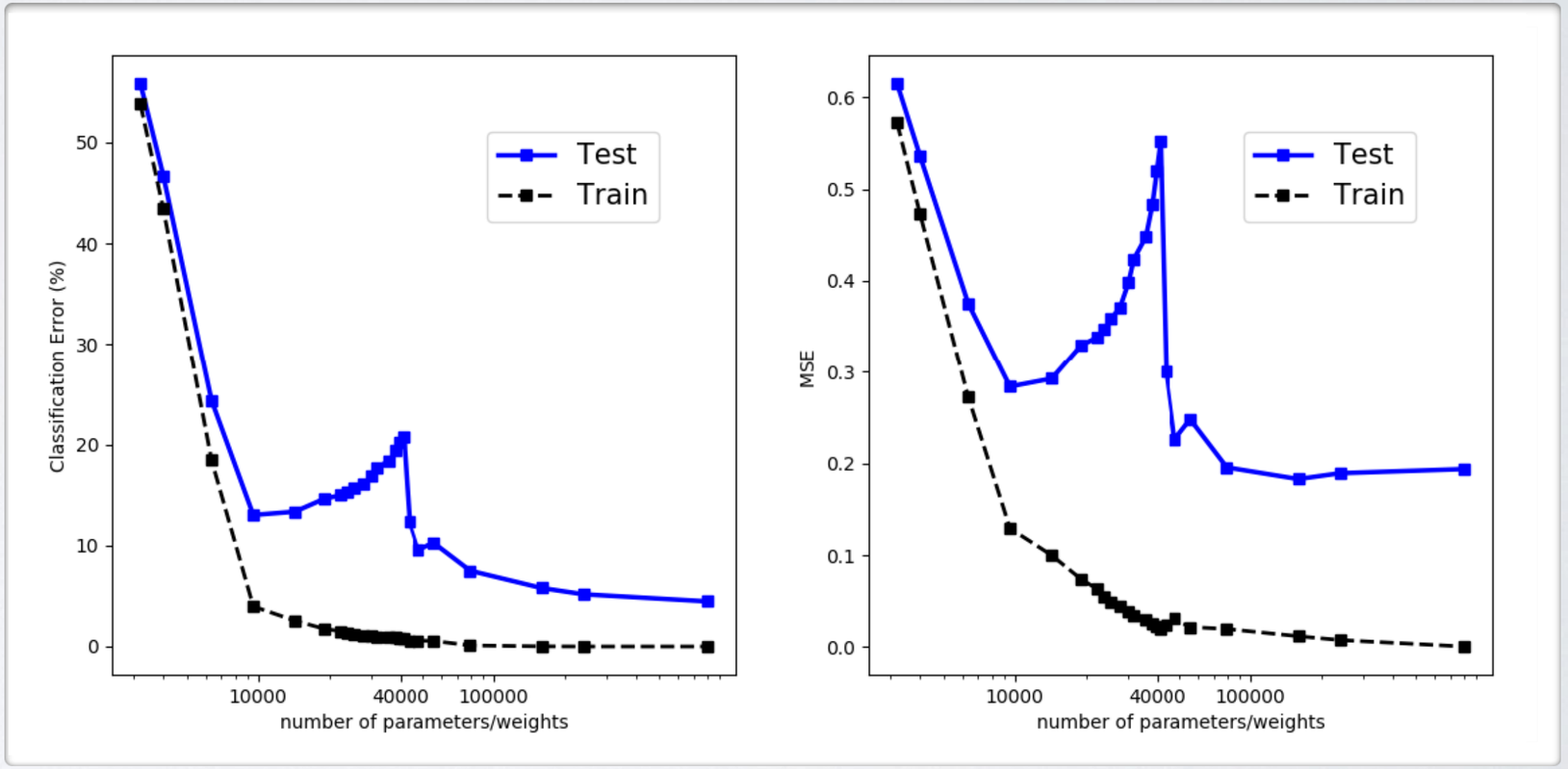
THEY OVERFIT STRANGELY

Topics: bias/variance trade-off, interpolation threshold

- *Reconciling modern machine learning and the bias-variance trade-off* Belkin et al. arXiv 2018

Interpretation(?):

- In *overparameterized NN, SDG finds a small norm solution, that leads to a smoother decision surface, subject to the fit of the data. Similar to (nonparametric) kernel methods.*



Regularization I

Parameter regularization

REGULARIZATION

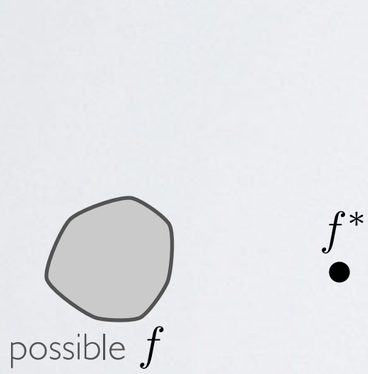
Topics: Why we regularize.

- How do we ensure our model will perform well not just on the training data, but also on new inputs?
  - i.e. How do we ensure **generalization**?
- One strategy: **Regularization (others?)**
- **Definition**: Putting extra constraints on a machine learning model, to improved performance on the **test set** (not training set), either by encoding prior knowledge into the model, or by forcing the model to consider alternative hypotheses that explain the training data.

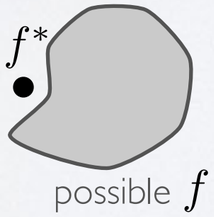
REGULARIZATION

Topics: bias-variance trade-off

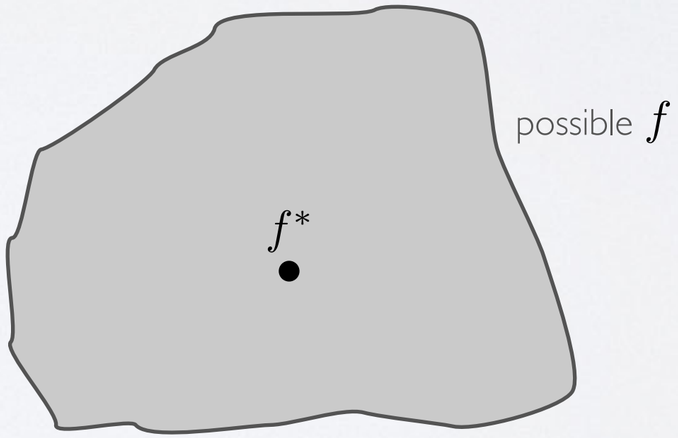
- Variance of trained model: does it vary a lot if the training set changes?
- Bias of trained model: is the average model close to the true solution?
- Generalization error can be seen as the sum of the (squared) bias and the variance (the mean squared error - MSE)



low variance/ high bias



good trade-off



high variance/ low bias

REGULARIZATION

Topics: Why we regularize.

- Regularizers trade increased bias for reduced variance.
- An effective regularizer is one that makes a profitable trade, that is, it reduces variance significantly while not overly increasing the bias.

REGULARIZATION

In the context of deep learning:

- Often working with data such as images, audio sequences and text:
  - we can safely assume that the model family we are training does not include the data generating process.
- Complexity of the model is not about finding the model of the right size. i.e. the right number of parameters.
- Instead, the best fitting model is one that possesses a large number of parameters that are not entirely free to span their domain.

REGULARIZATION

Recall: **Structured risk** (loss function we are minimizing)

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

Empirical Risk

Regularization Term

- α is a hyperparameter that balances the relative effect of the regularization term.
- For neural network models, we typically have $\boldsymbol{\theta} = [\boldsymbol{w}^\top, b]^\top$
- Typical regularizers: $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$ or $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

Taking gradient

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha\boldsymbol{w} + \nabla_{\boldsymbol{w}}J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{w})$$

We are going to gain some insight into how this works!

REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

- Consider a quadratic approximation to the loss function in the neighbourhood of the empirically optimal value of the weights $\boldsymbol{w}^*$

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

Taking gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

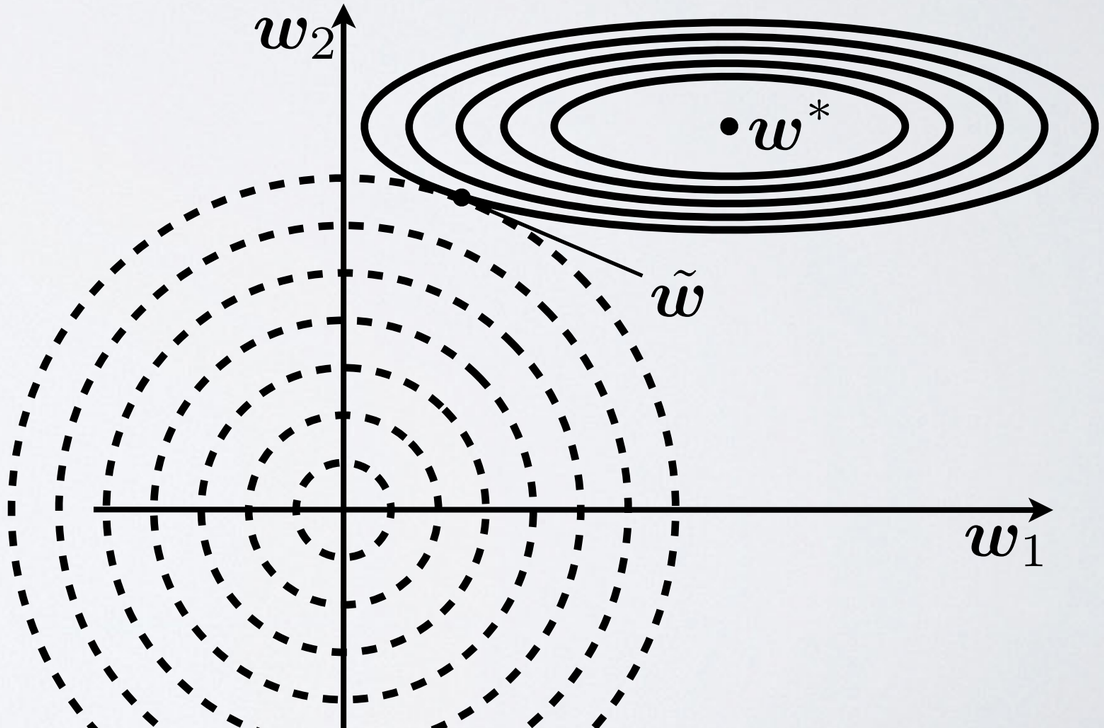
$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

- Consider the effect of L2 regularization around the unregularized optimum $\boldsymbol{w}^*$

$$\alpha \boldsymbol{w} + \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*) = 0$$

$$(\boldsymbol{H} + \alpha \boldsymbol{I})\boldsymbol{w} = \boldsymbol{H}\boldsymbol{w}^*$$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{w}^*$$



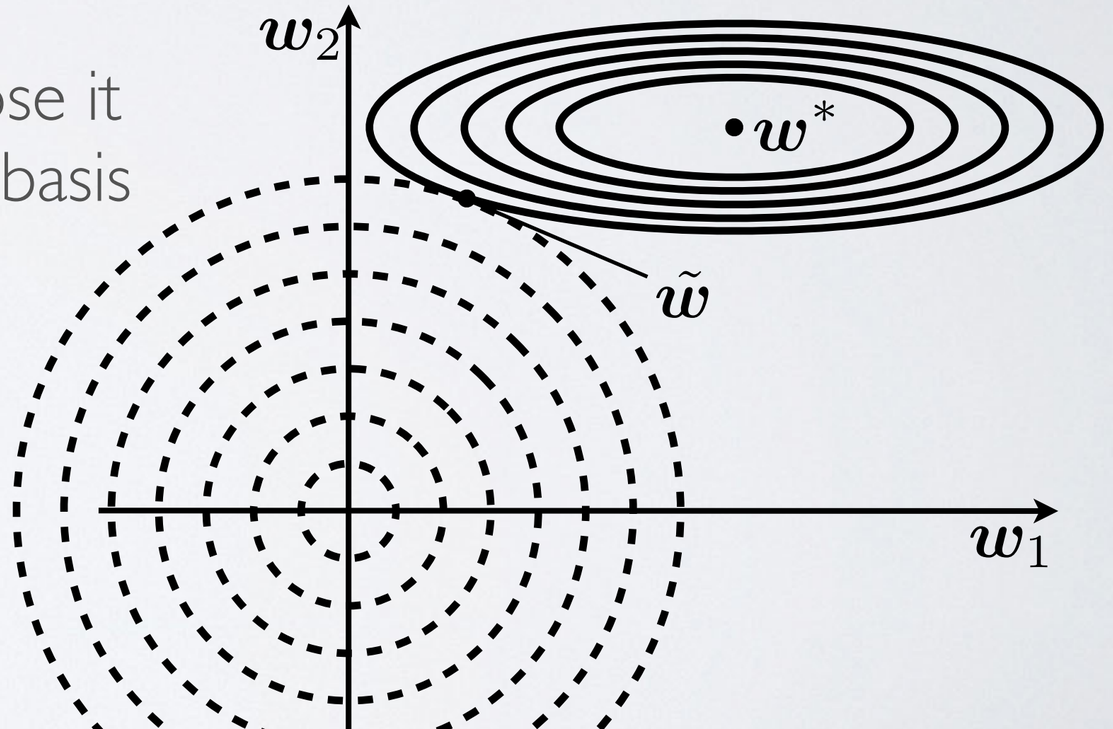
REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{w}^*$$

- The presence of the regularization term moves the optimum from $\boldsymbol{w}^*$ to $\tilde{\boldsymbol{w}}$
- $\boldsymbol{H}$ is real and symmetric, so we can decompose it into a diagonal matrix $\boldsymbol{\Lambda}$ and an orthogonal basis of eigenvectors, $\boldsymbol{Q}$, such that:

$$\boldsymbol{H} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top$$



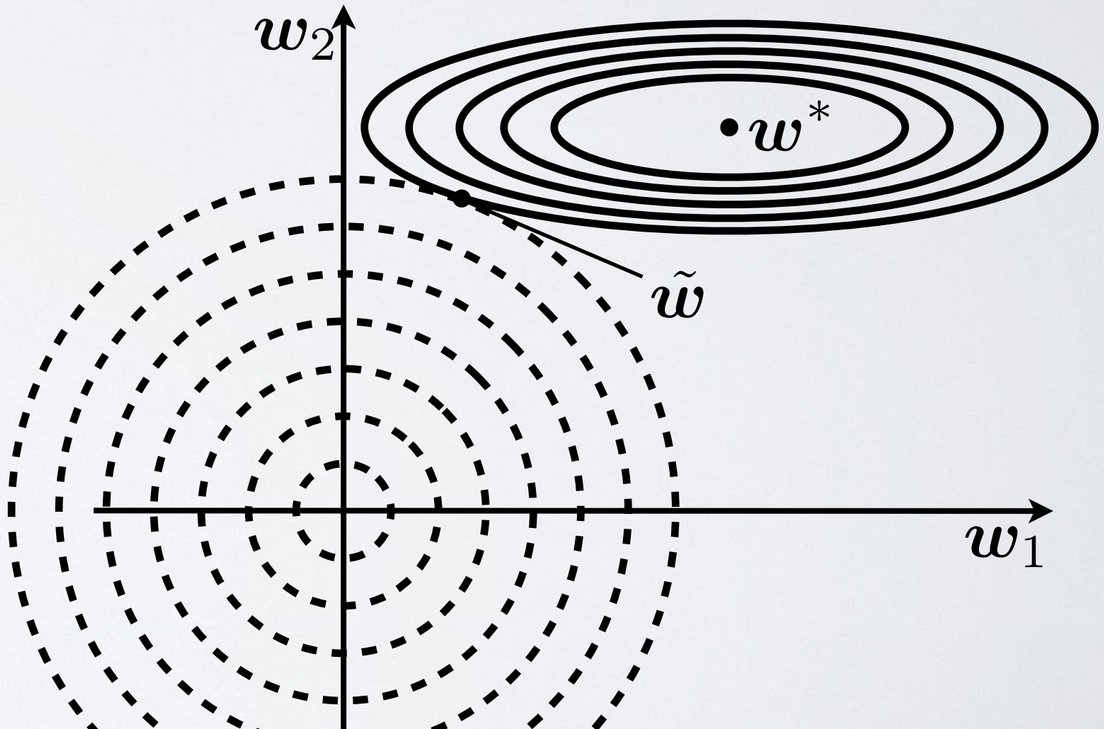
REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{w}^*$$

- In the above, if we swap in $\boldsymbol{H} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top$ we get:

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^*$$



REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1} \boldsymbol{H} \boldsymbol{w}^*$$

- In the above, if we swap in $\boldsymbol{H} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^\top$

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^*$$

Why?

$$\begin{aligned}
\tilde{w} &= (H + \alpha I)^{-1} H w^* \\
&= (Q \Lambda Q^\top + \alpha I)^{-1} Q \Lambda Q^\top w^* \\
&= \left(Q(\Lambda + \alpha I) Q^\top\right)^{-1} Q \Lambda Q^\top w^* \\
&= Q \left(\Lambda + \alpha I\right)^{-1} Q^\top Q \Lambda Q^\top w^* \\
&= Q \left(\Lambda + \alpha I\right)^{-1} \Lambda Q^\top w^* \\
Q^\top \tilde{w} &= Q^\top Q \left(\Lambda + \alpha I\right)^{-1} \Lambda Q^\top w^* \\
&= \left(\Lambda + \alpha I\right)^{-1} \Lambda Q^\top w^*
\end{aligned}$$

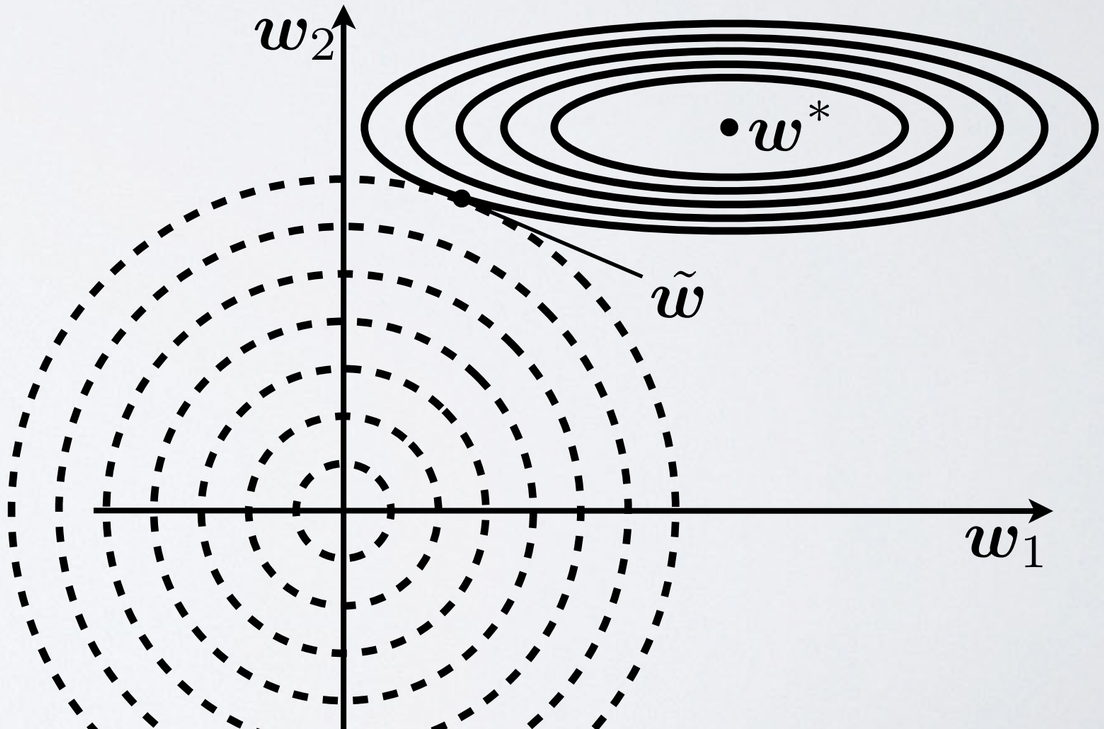
REGULARIZATION

Topics: L2 regularization $\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2$

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{Q}^\top \boldsymbol{w}^*$$

- The different components of $\boldsymbol{w}^*$ are rescaled by the regularization.
- The component aligned with eigenvector i is rescaled by a factor

$$\frac{\lambda_i}{\lambda_i + \alpha}$$



REGULARIZATION

Topics: L1 regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \beta\Omega(\boldsymbol{\theta})$$

Taking gradient

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \nabla_{\boldsymbol{w}} J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{w}) + \beta \text{sign}(\boldsymbol{w})$$

REGULARIZATION

Topics: L1 regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \beta\Omega(\boldsymbol{\theta})$$

- Consider a quadratic approximation to the loss function in the neighbourhood of the empirically optimal value of the weights $\boldsymbol{w}^*$

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

Taking gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

REGULARIZATION

Topics: L1 regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

Taking gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

- We will also make the further simplifying assumption that the Hessian is diagonal, $\boldsymbol{H} = \mathrm{diag}([\gamma_1, \ldots, \gamma_N])$, where each $\gamma_i > 0$

REGULARIZATION

Topics: L1 regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

- Under these assumptions the objective simplifies to a system of equations:

$$\tilde{J}(\boldsymbol{w}_i; \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{2}\gamma_i(\boldsymbol{w}_i - \boldsymbol{w}_i^*)^2 + \beta|\boldsymbol{w}_i|.$$

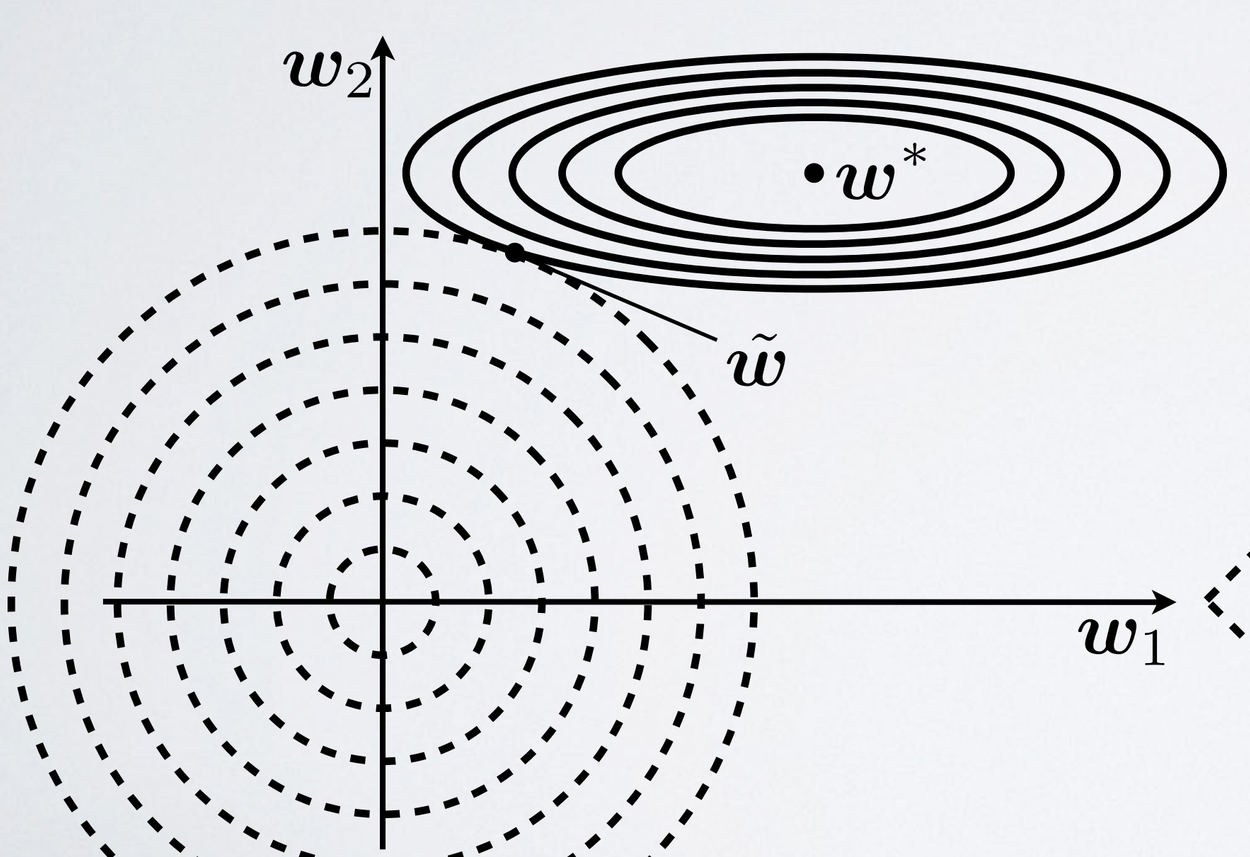
- Which admits an optimal solution (for each dimension) in the following form:

$$\boldsymbol{w}_i = \text{sign}(\boldsymbol{w}_i^*)\max(|\boldsymbol{w}_i^*| - \frac{\beta}{\gamma_i}, 0)$$

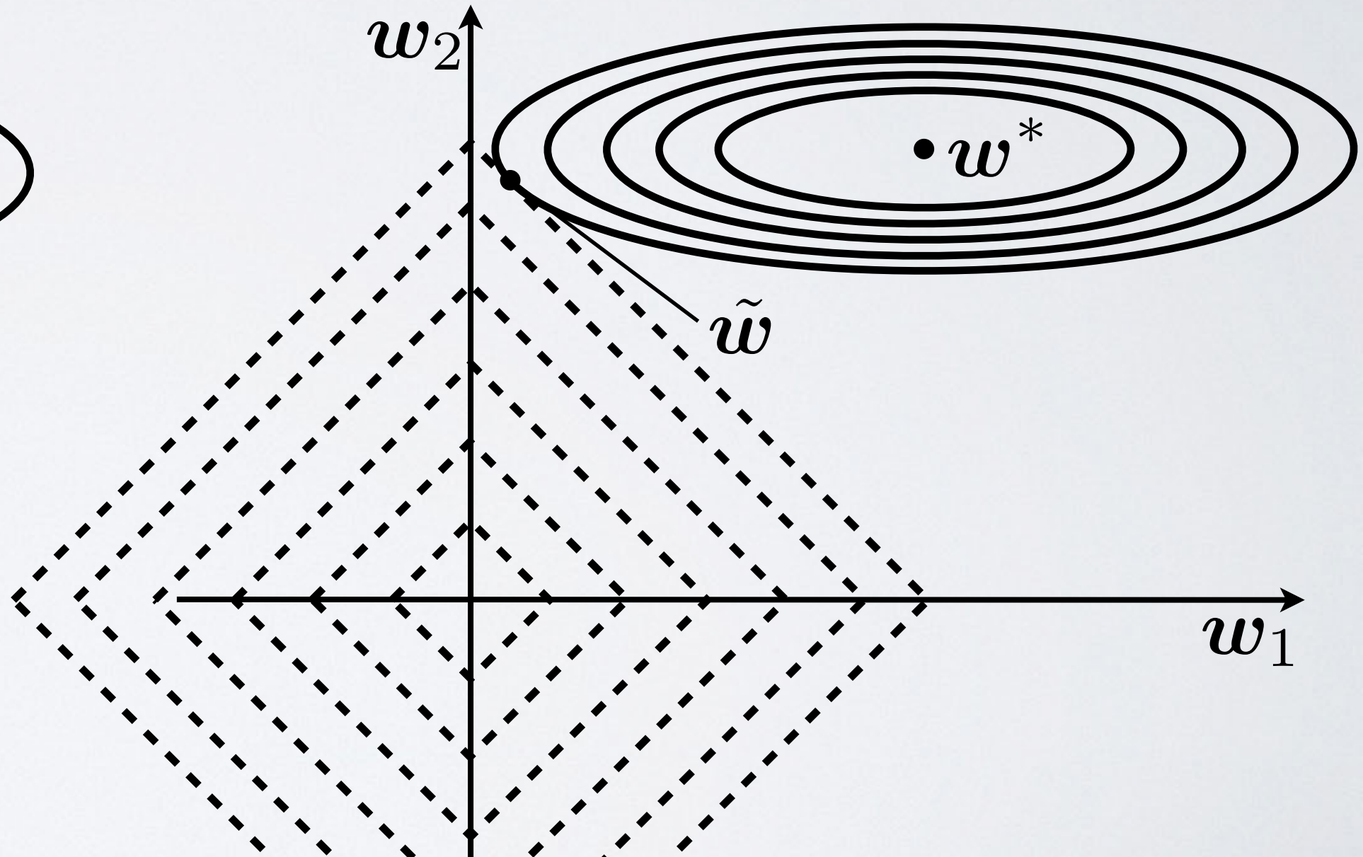
REGULARIZATION

Topics: L1 regularization $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1$

L2 regularization



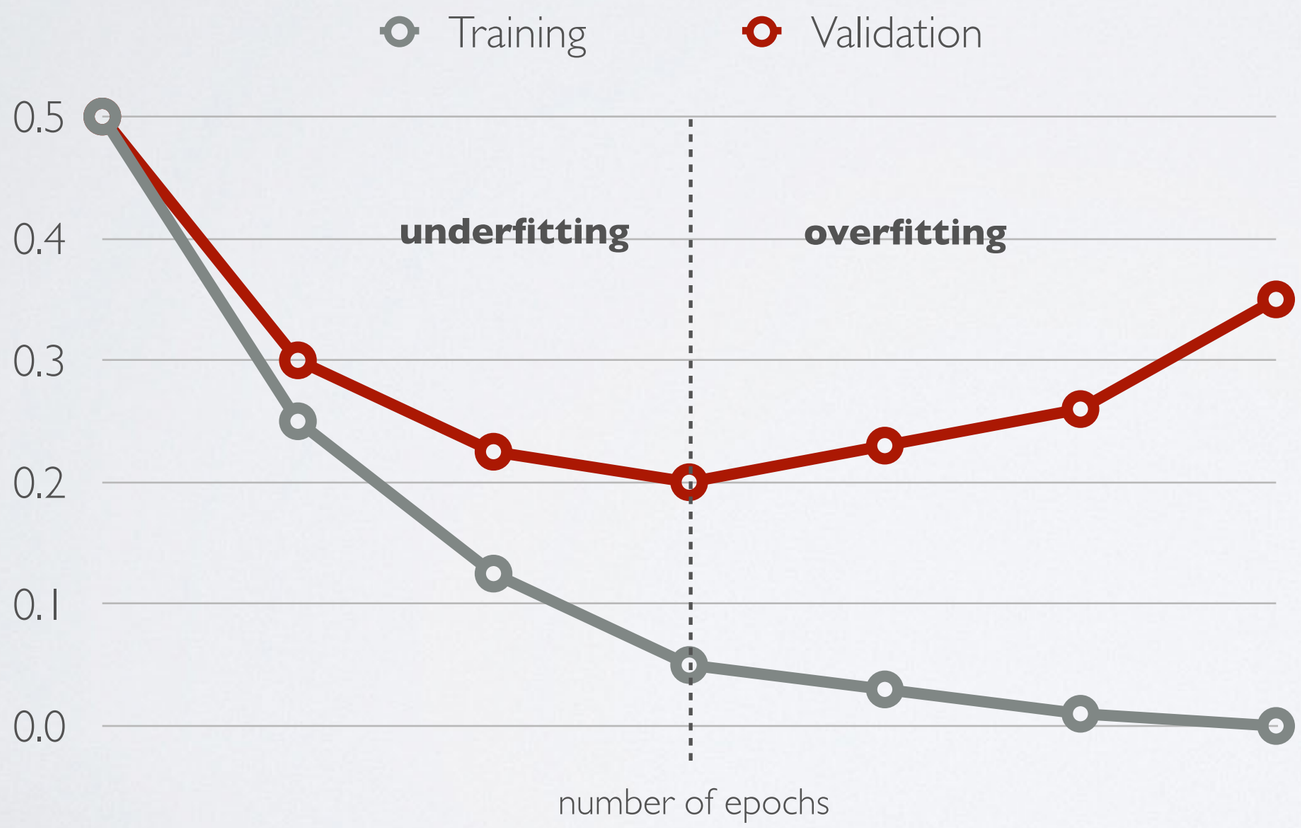
L1 regularization



KNOWING WHEN TO STOP

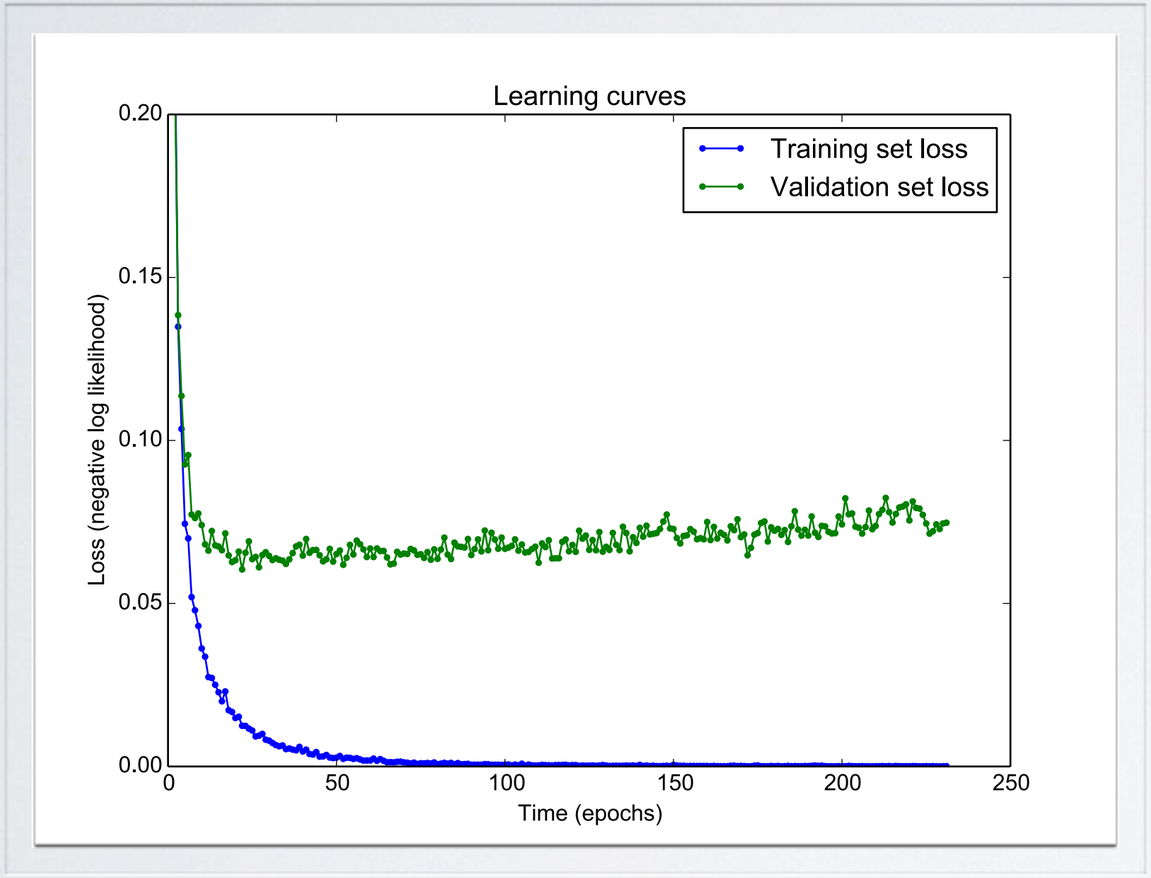
Topics: early stopping

- To select the number of epochs, stop training when validation set error increases (with some look ahead)



REGULARIZATION

Topics: Early stopping in practice



Algorithm 1 The early stopping meta-algorithm for determining the best amount of time to train. This meta-algorithm is a general strategy that works well with a variety of training algorithms and ways of quantifying error on the validation set.

Let n be the number of steps between evaluations.
Let p be the "patience," the number of times to observe worsening validation set error before giving up.
Let $\boldsymbol{\theta}_o$ be the initial parameters.
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_o$
$i \leftarrow 0$
$j \leftarrow 0$
$v \leftarrow \infty$
$\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
$i^* \leftarrow i$
while $j < p$ **do**
  Update $\boldsymbol{\theta}$ by running the training algorithm for n steps.
  $i \leftarrow i + n$
  $v' \leftarrow \text{ValidationSetError}(\boldsymbol{\theta})$
  if $v' < v$ **then**
    $j \leftarrow 0$
    $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
    $i^* \leftarrow i$
    $v \leftarrow v'$
  else
    $j \leftarrow j + 1$
  end if
end while
Best parameters are $\boldsymbol{\theta}^*$, best number of training steps is i^*

REGULARIZATION

Topics: Early stopping with retraining

- Sometimes you really don't want to "waste" the validation set by not training on it.
- There are two basic strategies for retraining with the validation data.

1. Retrain with train+valid for the same number of (updates / epochs) as determined by initial early stopping.
2. Continue training w/ train+valid until the loss on valid = early-stopped loss on train. Not guaranteed to stop.

Algorithm 1 A meta-algorithm for using early stopping to determine how long to train, then retraining on all the data.

Let $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ be the training set
Split $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ into $\boldsymbol{X}^{(\text{subtrain})}$, $\boldsymbol{y}^{(\text{subtrain})}$, $\boldsymbol{X}^{(\text{valid})}$, $\boldsymbol{y}^{(\text{valid})}$
Run early stopping starting from random $\boldsymbol{\theta}$ using $\boldsymbol{X}^{(\text{subtrain})}$ and $\boldsymbol{y}^{(\text{subtrain})}$ for training data and $\boldsymbol{X}^{(\text{valid})}$ and $\boldsymbol{y}^{(\text{valid})}$ for validation data. This returns i^*, the optimal number of steps.
Set $\boldsymbol{\theta}$ to random values again
Train on $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ for i^* steps.

Algorithm 2 A meta-algorithm for using early stopping to determining at what objective value we start to overfit, then continuing training.

Let $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ be the training set
Split $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ into $\boldsymbol{X}^{(\text{subtrain})}$, $\boldsymbol{y}^{(\text{subtrain})}$, $\boldsymbol{X}^{(\text{valid})}$, $\boldsymbol{y}^{(\text{valid})}$
Run early stopping (Alg. **??**) starting from random $\boldsymbol{\theta}$ using $\boldsymbol{X}^{(\text{subtrain})}$ and $\boldsymbol{y}^{(\text{subtrain})}$ for training data and $\boldsymbol{X}^{(\text{valid})}$ and $\boldsymbol{y}^{(\text{valid})}$ for validation data. This updates $\boldsymbol{\theta}$
$\epsilon \leftarrow J(\boldsymbol{\theta}, \boldsymbol{X}^{(\text{subtrain})}, \boldsymbol{y}^{(\text{subtrain})})$
while $J(\boldsymbol{\theta}, \boldsymbol{X}^{(\text{valid})}, \boldsymbol{y}^{(\text{valid})}) > \epsilon$ **do**
  Train on $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ for n steps.
end while

Warning: these methods are dangerous!

REGULARIZATION

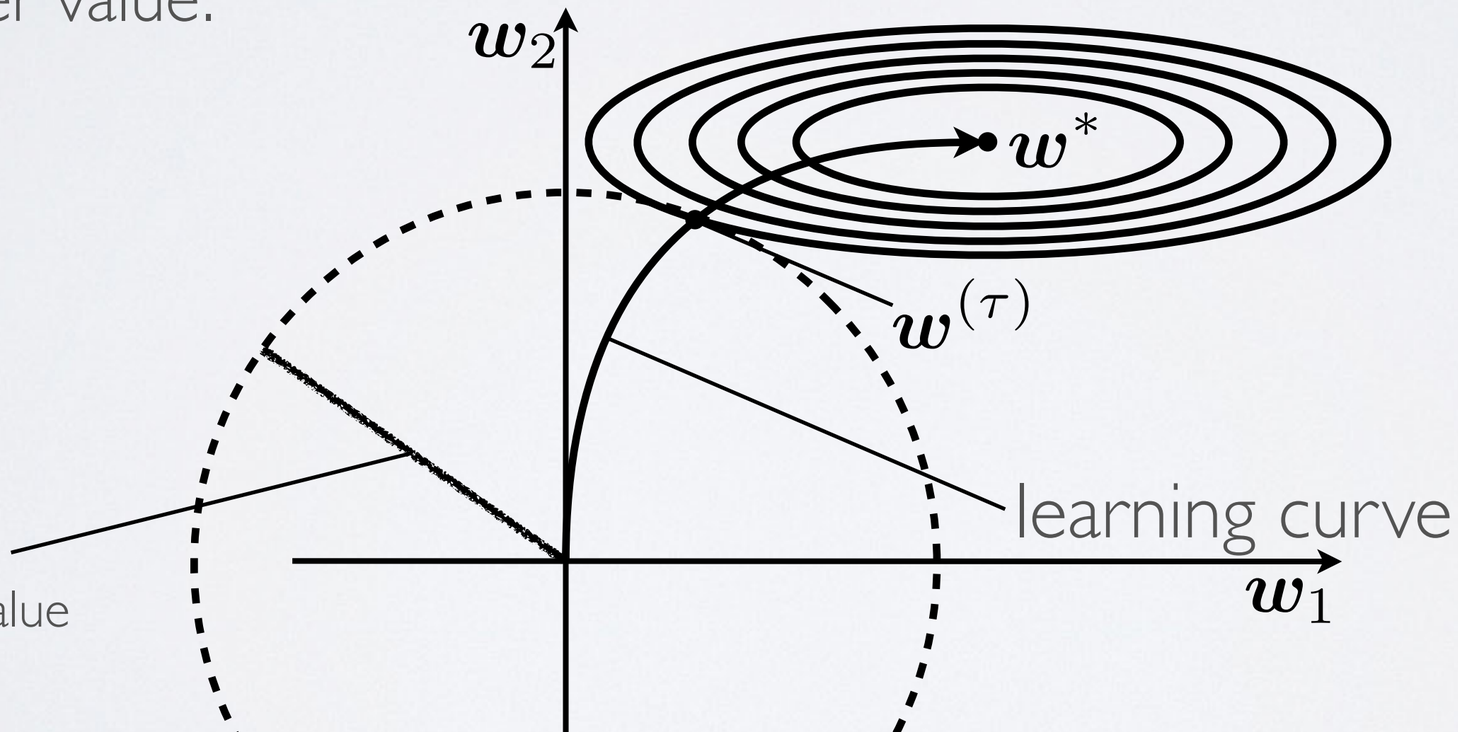
Topics: Early stopping with surrogate loss

- A useful property: can help to mitigate a mismatch between the surrogate loss and the underlying performance measure that we actually care about.
  - Example, 0-1 classification loss (derivative of zero almost everywhere). We therefore train with surrogates such as the log likelihood of correct class label.
  - However, 0-1 loss is inexpensive to compute, so it can easily be used as an early stopping criterion.
  - Often the 0-1 loss decreases long after the log likelihood has begun to worsen on the validation set.

REGULARIZATION

Topics: How early stopping acts as a regularizer.

- What is the actual mechanism by which early stopping regularizes the model?
  - Early stopping has the effect of restricting the optimization procedure to a relatively small volume of parameter space in the neighbourhood of the initial parameter value.

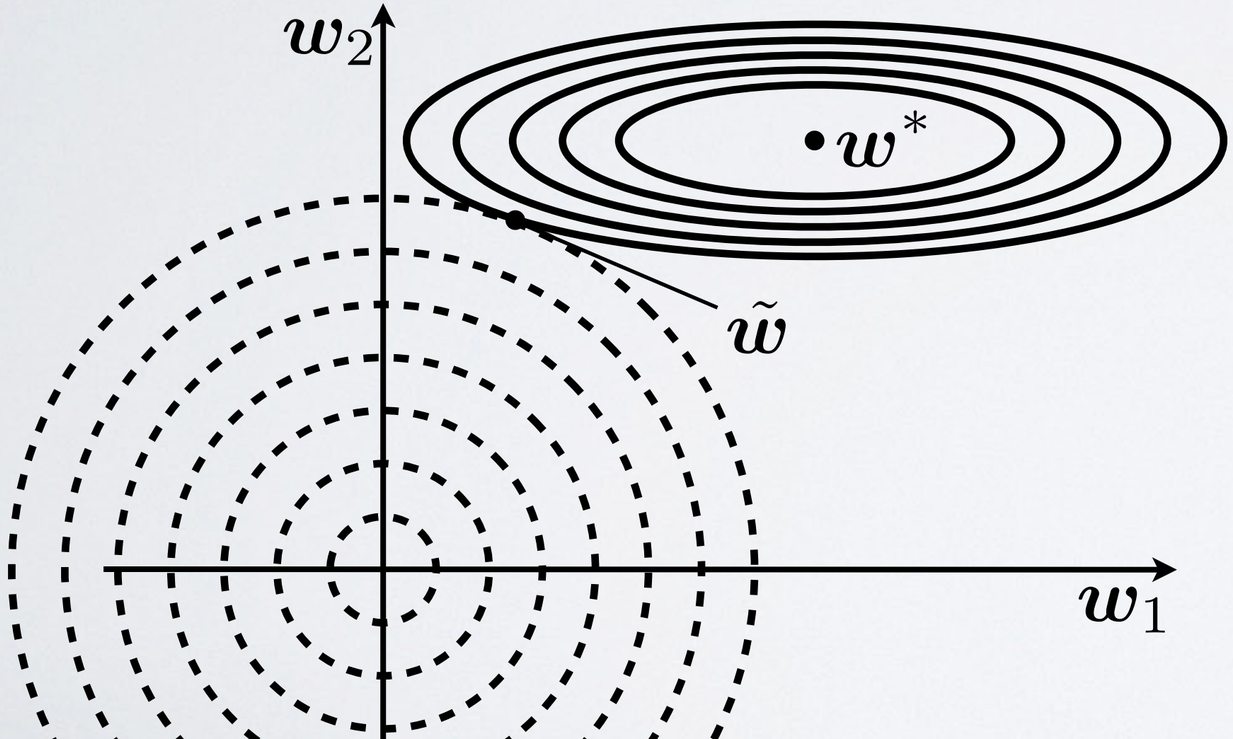


REGULARIZATION

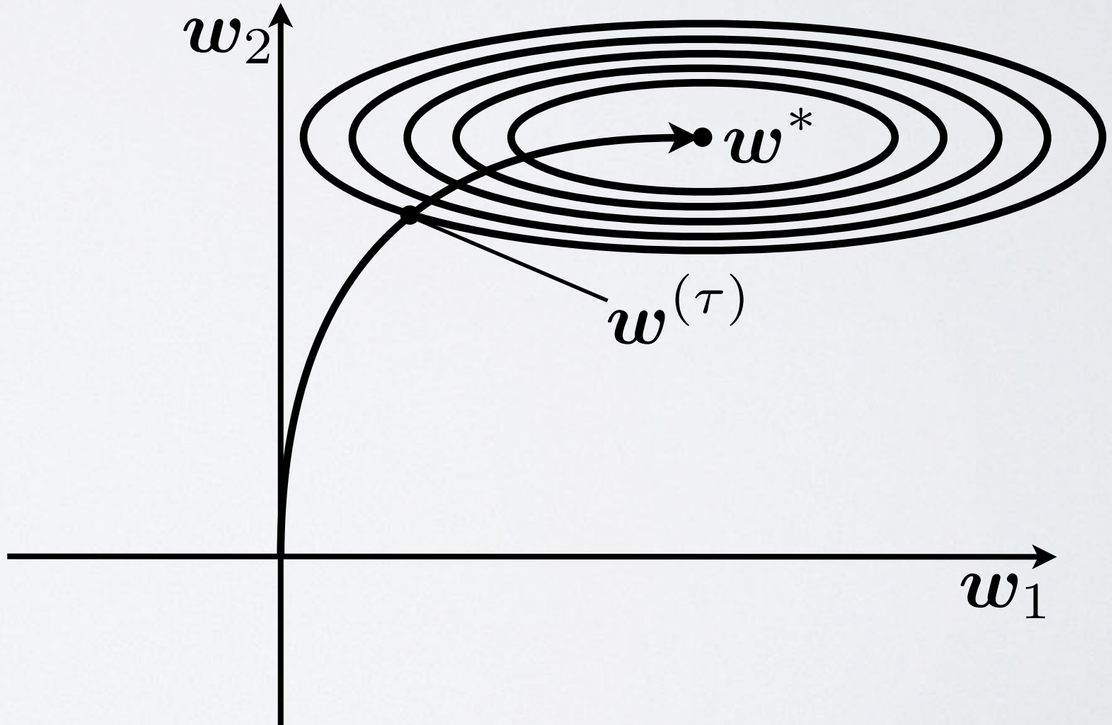
Topics: How early stopping acts as a regularizer.

- Assuming a simple linear model with a quadratic error function and simple gradient descent -- early stopping is equivalent to L2 regularization.

L2 regularization



Early Stopping



REGULARIZATION

Topics: Early stopping equivalence to L2 regularization, **mathematical details**.

- Consider a quadratic approximation to the loss function in the neighbourhood of the empirically optimal value of the weights $\boldsymbol{w}^*$

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

Taking gradient

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

REGULARIZATION

Topics: Early stopping equivalence to L2 regularization, **mathematical details**.

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

- Let us consider initial parameter vector chosen at the origin,
- We will consider updating the parameters via gradient descent:

$$\begin{aligned}
\boldsymbol{w}^{(\tau)} &= \boldsymbol{w}^{(\tau-1)} - \eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w}^{(\tau-1)}) \\
&= \boldsymbol{w}^{(\tau-1)} - \eta \boldsymbol{H}(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*) \\
\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* &= (\boldsymbol{I} - \eta \boldsymbol{H})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*)
\end{aligned}$$

REGULARIZATION

Topics: Early stopping equivalence to L2 regularization, **mathematical details**.

$$\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* = (\boldsymbol{I} - \eta \boldsymbol{H})(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*)$$

- $\boldsymbol{H}$ is real and symmetric, so we can decompose it into a diagonal matrix $\boldsymbol{\Lambda}$ and an orthogonal basis of eigenvectors, $\boldsymbol{Q}$, such that: $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$

$$\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^* = (\boldsymbol{I} - \eta \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top)(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*)$$

$$\boldsymbol{Q}^\top(\boldsymbol{w}^{(\tau)} - \boldsymbol{w}^*) = (\boldsymbol{I} - \eta \boldsymbol{\Lambda})\boldsymbol{Q}^\top(\boldsymbol{w}^{(\tau-1)} - \boldsymbol{w}^*)$$

- Assuming that $|1 - \eta\lambda_i| < 1$ and that $\boldsymbol{w}^{(0)} = \boldsymbol{0}$. After τ steps:

$$\boldsymbol{Q}^\top \boldsymbol{w}^{(\tau)} = [\boldsymbol{I} - (\boldsymbol{I} - \eta\boldsymbol{\Lambda})^\tau]\boldsymbol{Q}^\top \boldsymbol{w}^*$$

REGULARIZATION

Topics: Early stopping equivalence to L2 regularization, **mathematical details**.

$$\boldsymbol{Q}^\top \boldsymbol{w}^{(\tau)} = [\boldsymbol{I} - (\boldsymbol{I} - \eta\boldsymbol{\Lambda})^\tau]\boldsymbol{Q}^\top \boldsymbol{w}^*$$

- Recall the L2 regularized solution was: $\tilde{\boldsymbol{w}} = \boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^*$

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = (\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^*$$

$$\boldsymbol{Q}^\top \tilde{\boldsymbol{w}} = [\boldsymbol{I} - (\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\alpha]\boldsymbol{Q}^\top \boldsymbol{w}^*$$

- These are **equivalent** when $(\boldsymbol{I} - \eta\boldsymbol{\Lambda})^\tau = (\boldsymbol{\Lambda} + \alpha\boldsymbol{I})^{-1}\alpha$

$$\tau \log(\boldsymbol{I} - \eta\boldsymbol{\Lambda}) = -\log(\boldsymbol{I} + \boldsymbol{\Lambda}/\alpha)$$

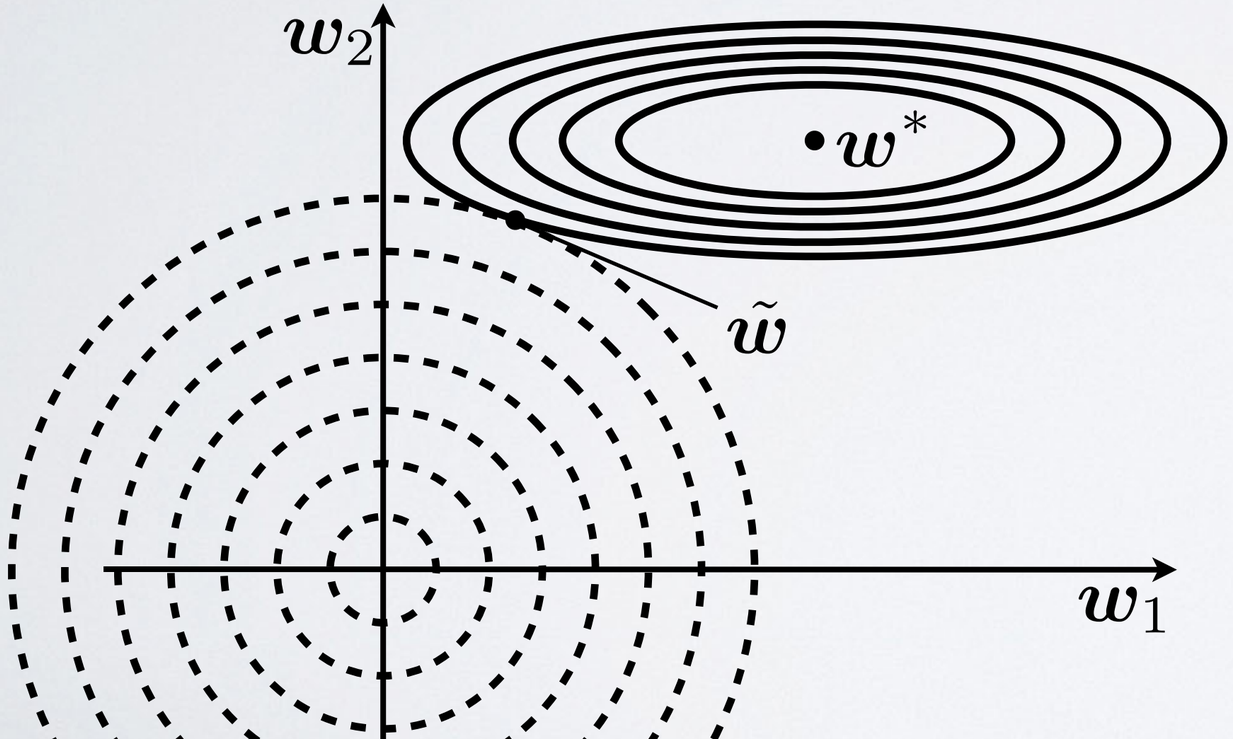
(by Taylor series expansion) $\qquad \tau \approx 1/\eta\alpha \quad \text{for small } \lambda_i \ \forall i$

REGULARIZATION

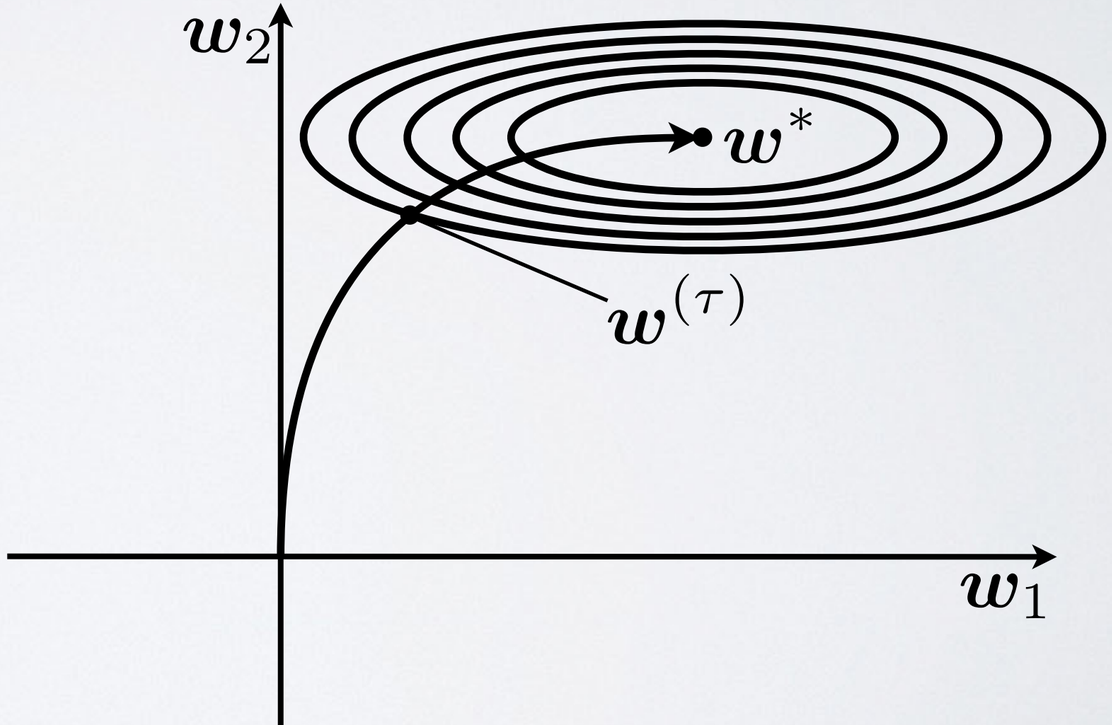
Topics: How early stopping acts as a regularizer.

- Assuming a simple linear model with a quadratic error function and simple gradient descent -- early stopping is equivalent to L2 regularization.

L2 regularization



Early Stopping



Dropout Training

(Hinton et al. 2012)

Aaron Courville

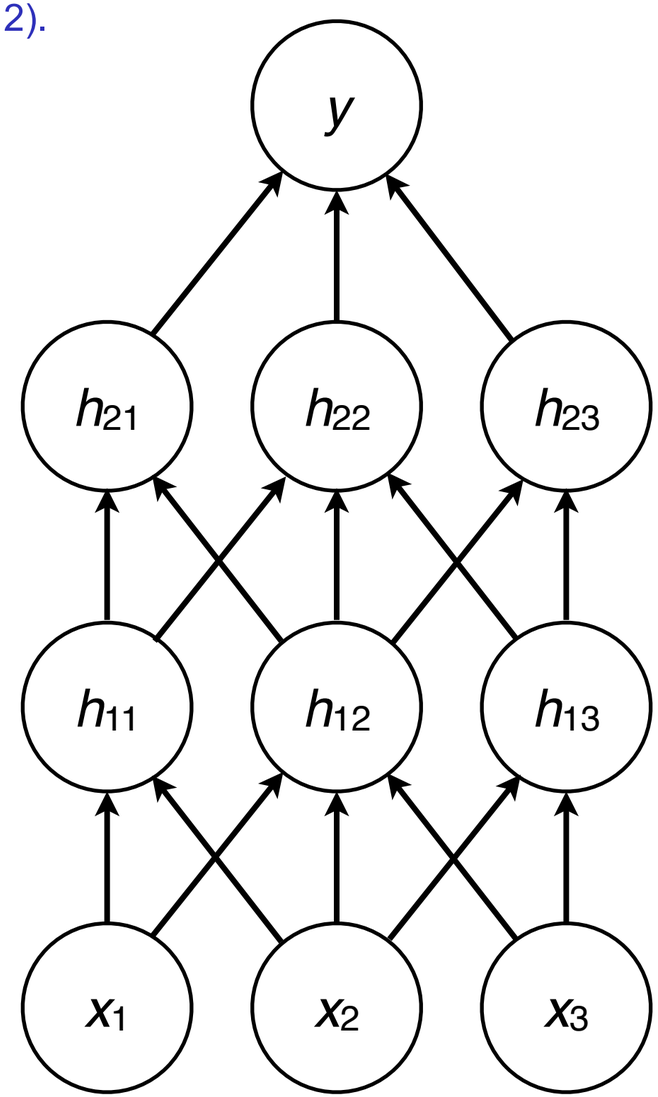
IFT6135 - Representation Learning

Slide Credit: Some slides were taken from Ian Goodfellow

Dropout training

- Introduced in Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.CoRR, abs/1207.0580.

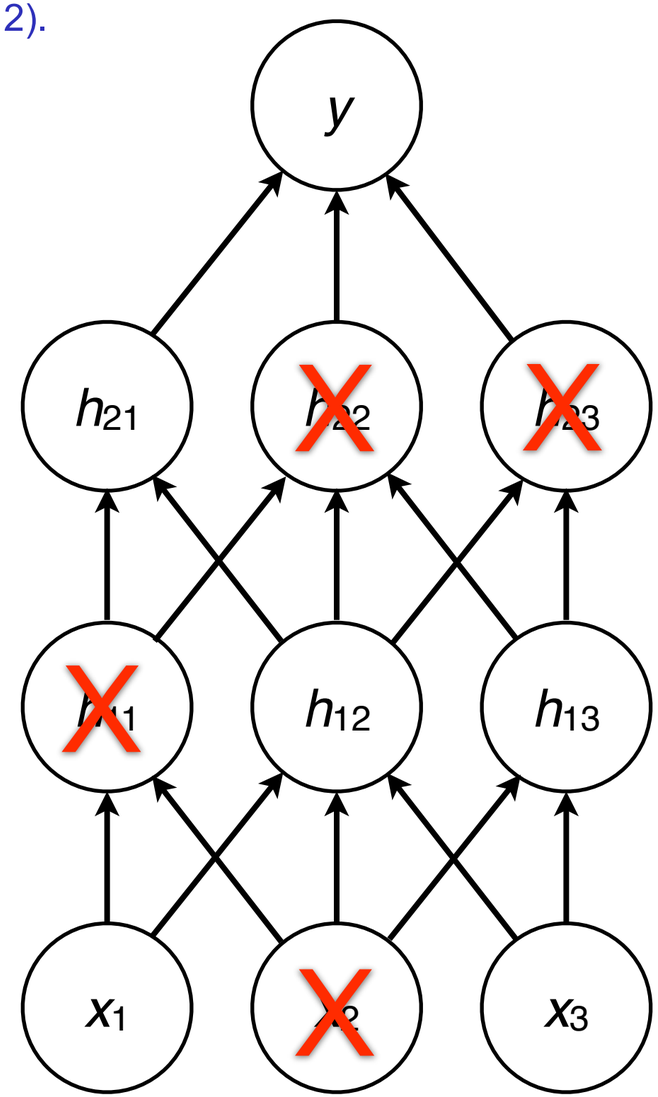
- **Dropout recipe**:
  - Each time we present data example ***x***, randomly delete each hidden node with 0.5 probability.
  - This is like sampling from $2^{|h|}$ different architectures.
  - At test time, use all nodes but divide the weights by 2.
- **Effect I**: Reduce overfitting by preventing ”co-adaptation”
- **Effect 2**: Ensemble model averaging via bagging



Dropout training

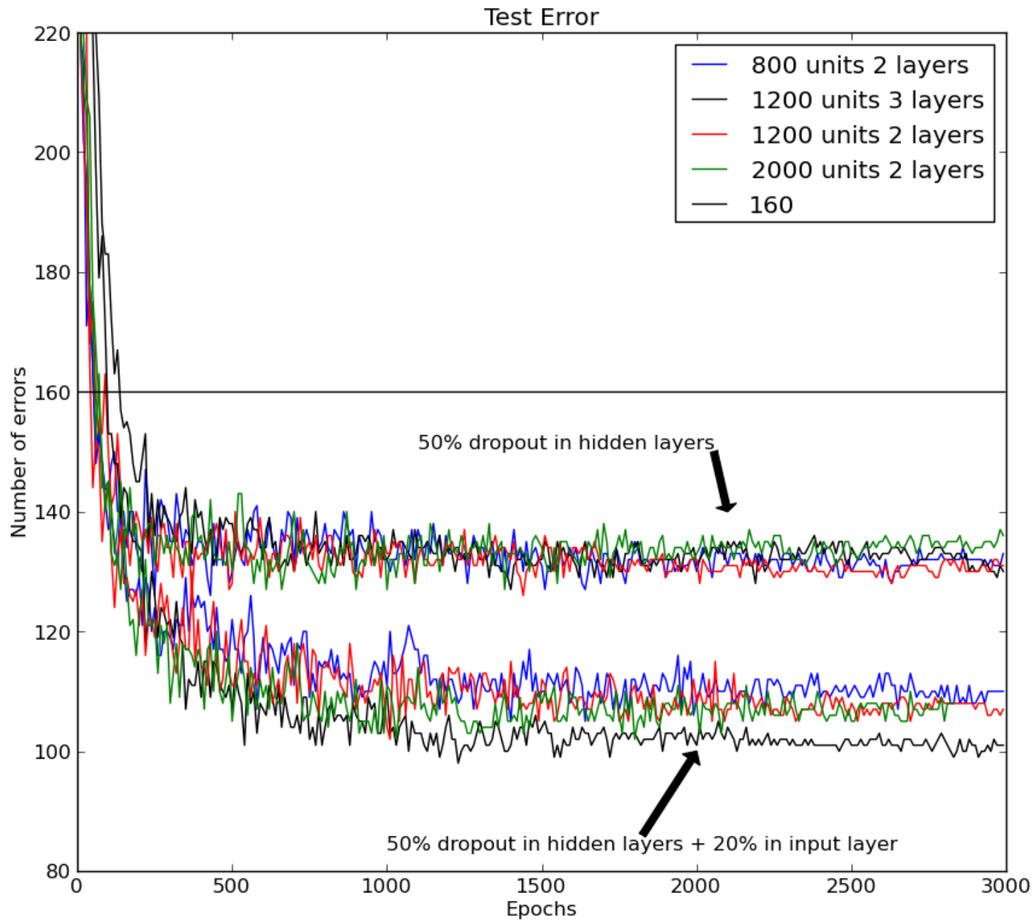
- Introduced in Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.CoRR, abs/1207.0580.

- **Dropout recipe**:
  - Each time we present data example ***x***, randomly delete each hidden node with 0.5 probability.
  - This is like sampling from $2^{|h|}$ different architectures.
  - At test time, use all nodes but divide the weights by 2.
- **Effect I**: Reduce overfitting by preventing ”co-adaptation”
- **Effect 2**: Ensemble model averaging via bagging



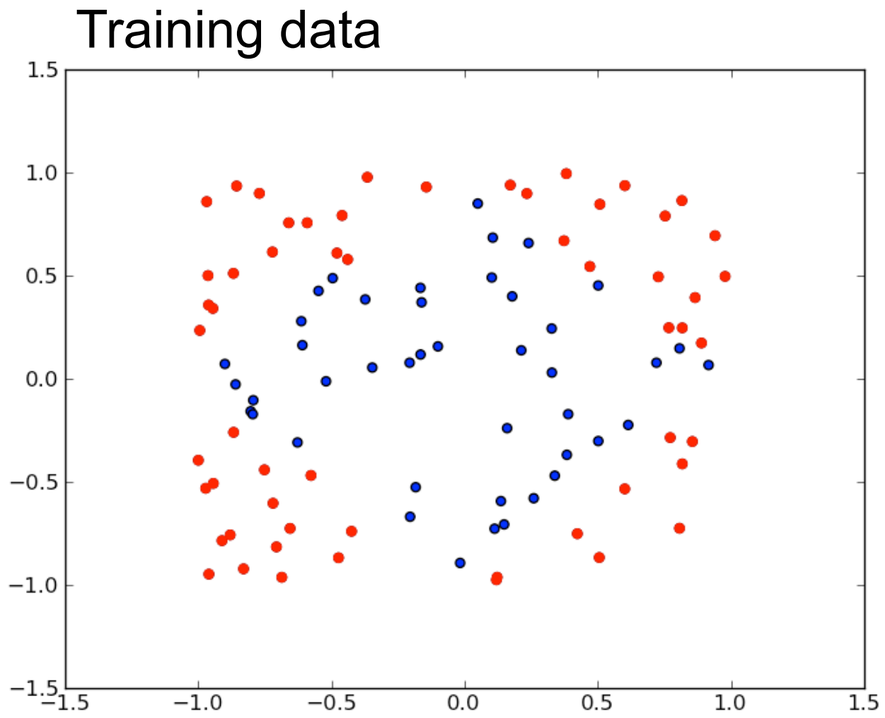
Dropout: MNIST digit recognition

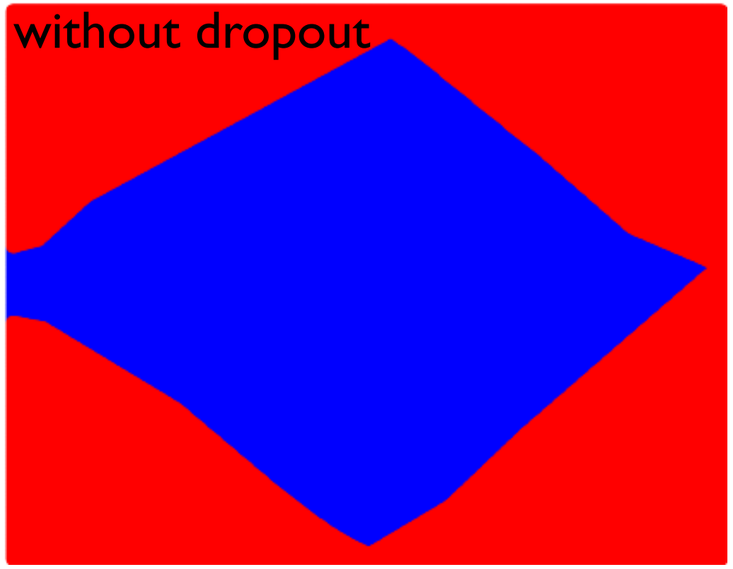
- Dropout is effective on MNIST.
- Particularly with input dropout.
- Comparison against other regularizers.

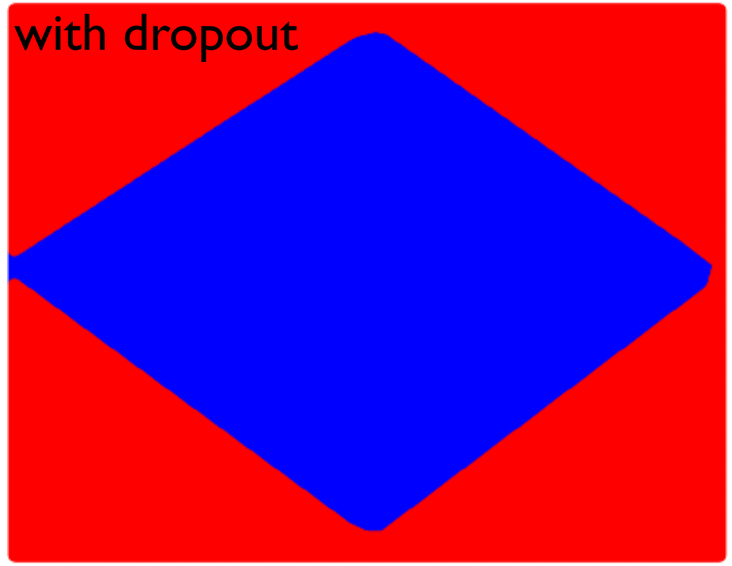


Method	MNIST Classification error %
L2	1.62
L1 (towards the end of training)	1.60
KL-sparsity	1.55
Max-norm	1.35
Dropout	1.25
Dropout + Max-norm	**1.05**

The unreasonable effectiveness of dropout







- A simple 2D example.
- Decision surfaces after training:

Claim: Dropout is approximate model averaging

- Hinton et al. (2012):
  - Dropout **approximates geometric model averaging**.

Arithmetic mean: $\frac{1}{N}\sum_{i=1}^{N} x_i$ Geometric mean: $\left(\prod_{i=1}^{N} x_i\right)^{\frac{1}{N}}$

Claim: Dropout is approximate model averaging

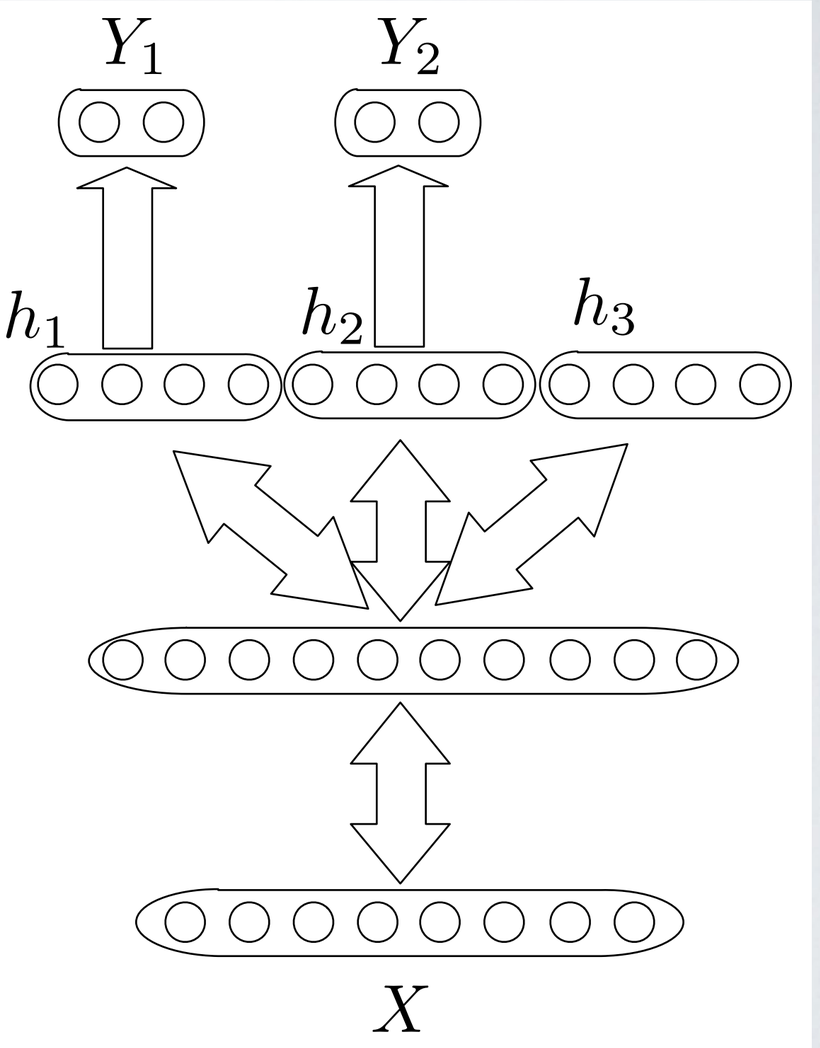
- In networks with a single hidden layer of N units and a “softmax” output layer:
- Using the “mean” network (using all weights/2) is exactly equivalent to taking the geometric mean of the probability distributions over labels predicted by all 2^N possible networks.
- For deep networks, it’s an approximation.

Promoting generalization by training on other (related) tasks.

MULTI-TASK LEARNING / UNSUPERVISED LEARNING / SELF-SUPERVISED LEARNING.

- Many strategies exist to leverage other related tasks to **regularize** the parameters of the target task:
  - unsupervised learning,
  - transfer learning
  - self-supervised learning

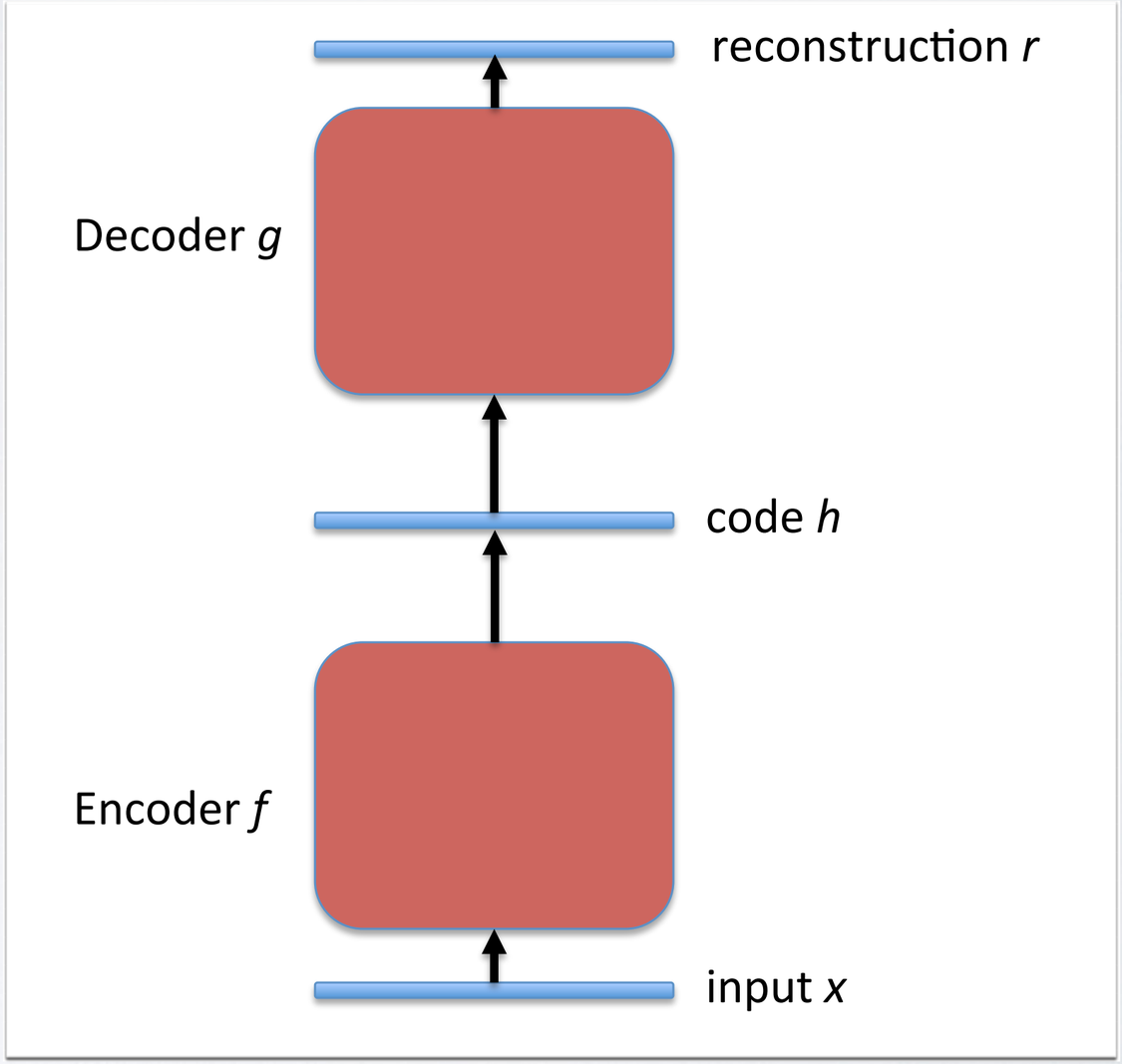
- True even when there are multiple target tasks as in multi-task learning.
  - Each task regularizers the others.



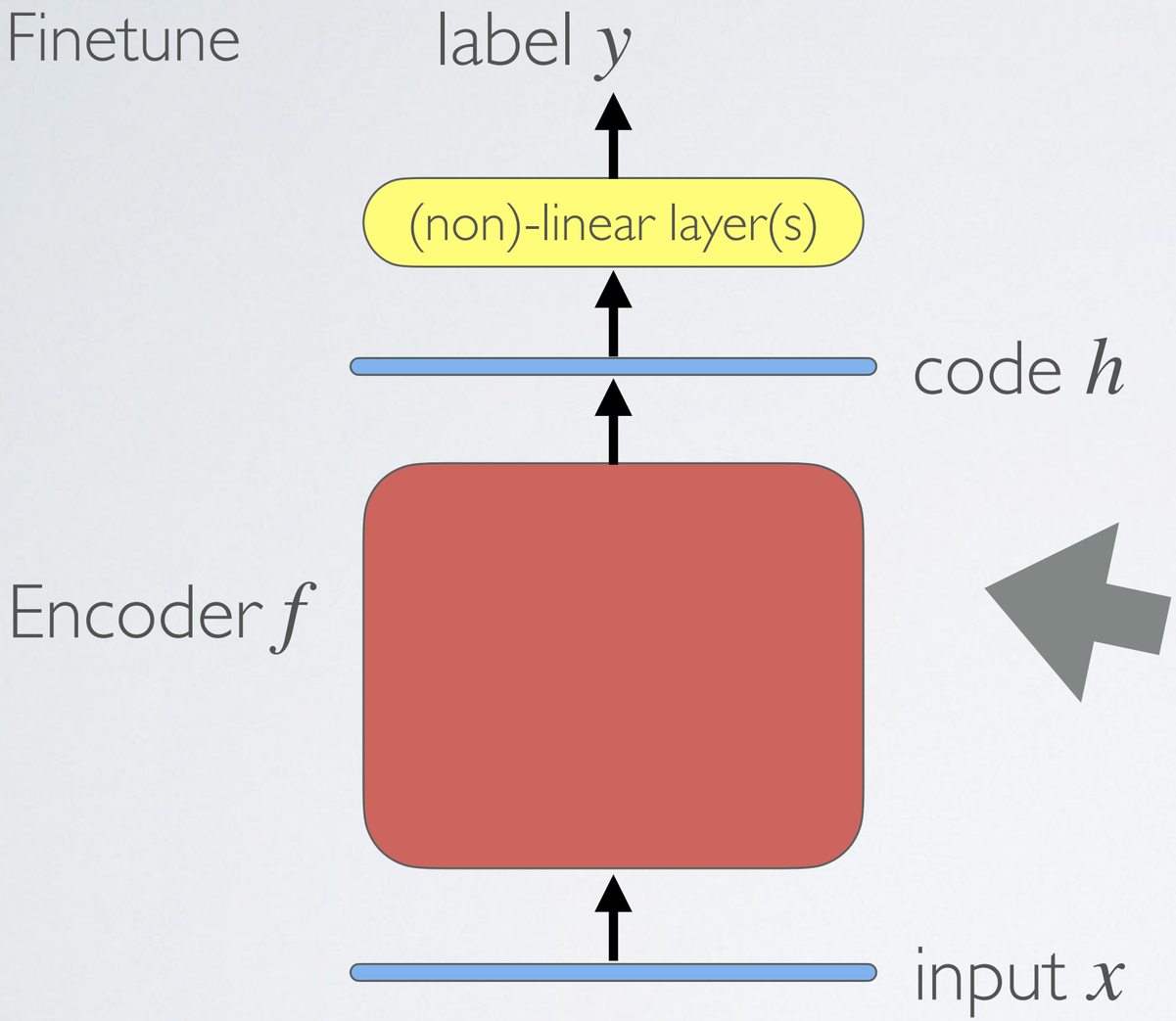
UNSUPERVISED LEARNING AS A REGULARIZATION STRATEGY

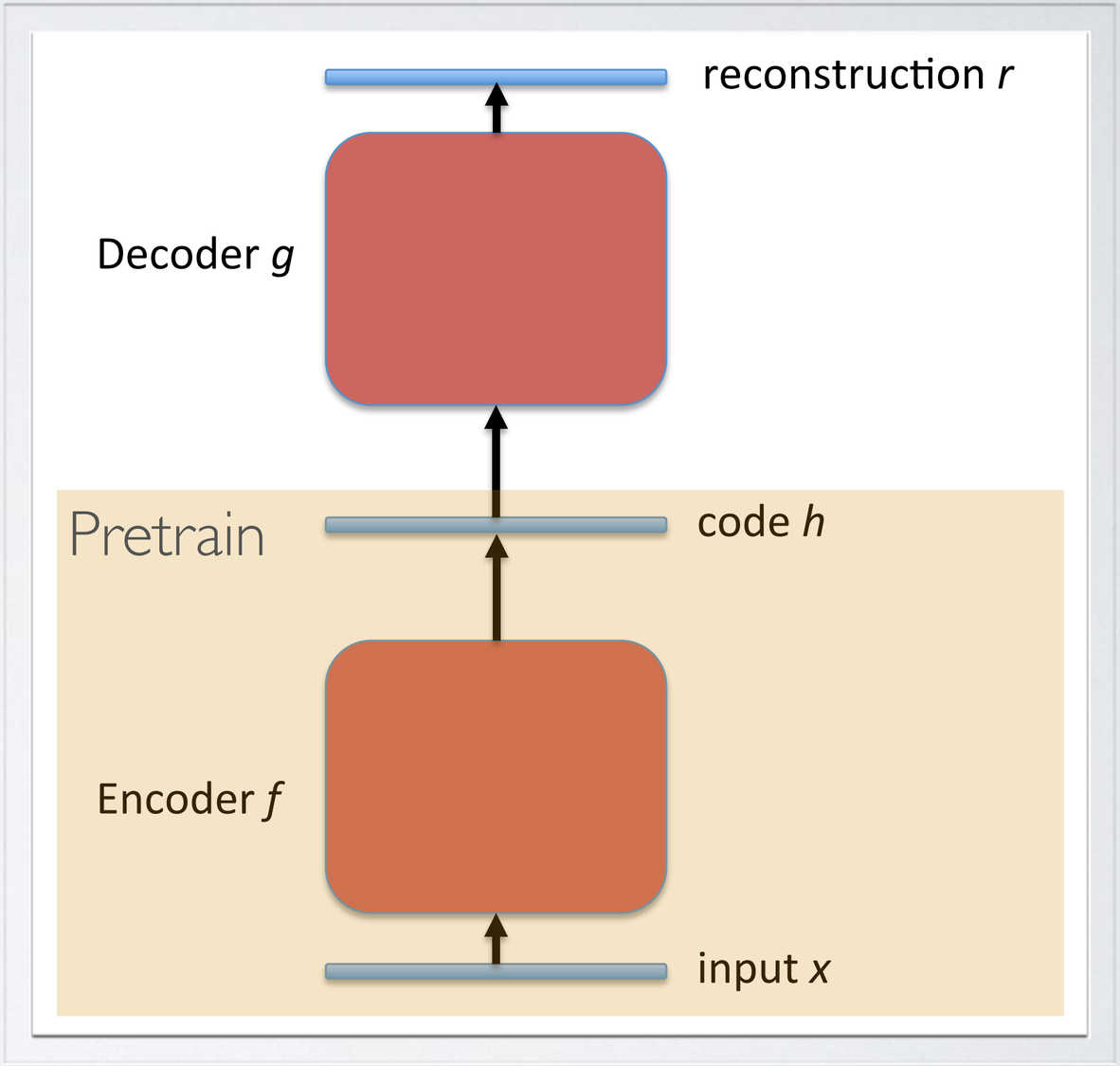
Topics: Autoencoders.

- Idea: pretrain some of discriminative model parameters as an autoencoder.
- Autoencoders used to feature prominently in the deep learning literature (less so now, but still popular)
- Goal: learn an encoder (f) and decoder (g) to minimize the error of reconstructing the input.
- Often, an additional penalty term is used to give the code (h) desirable characteristics (we will see this later in the course)



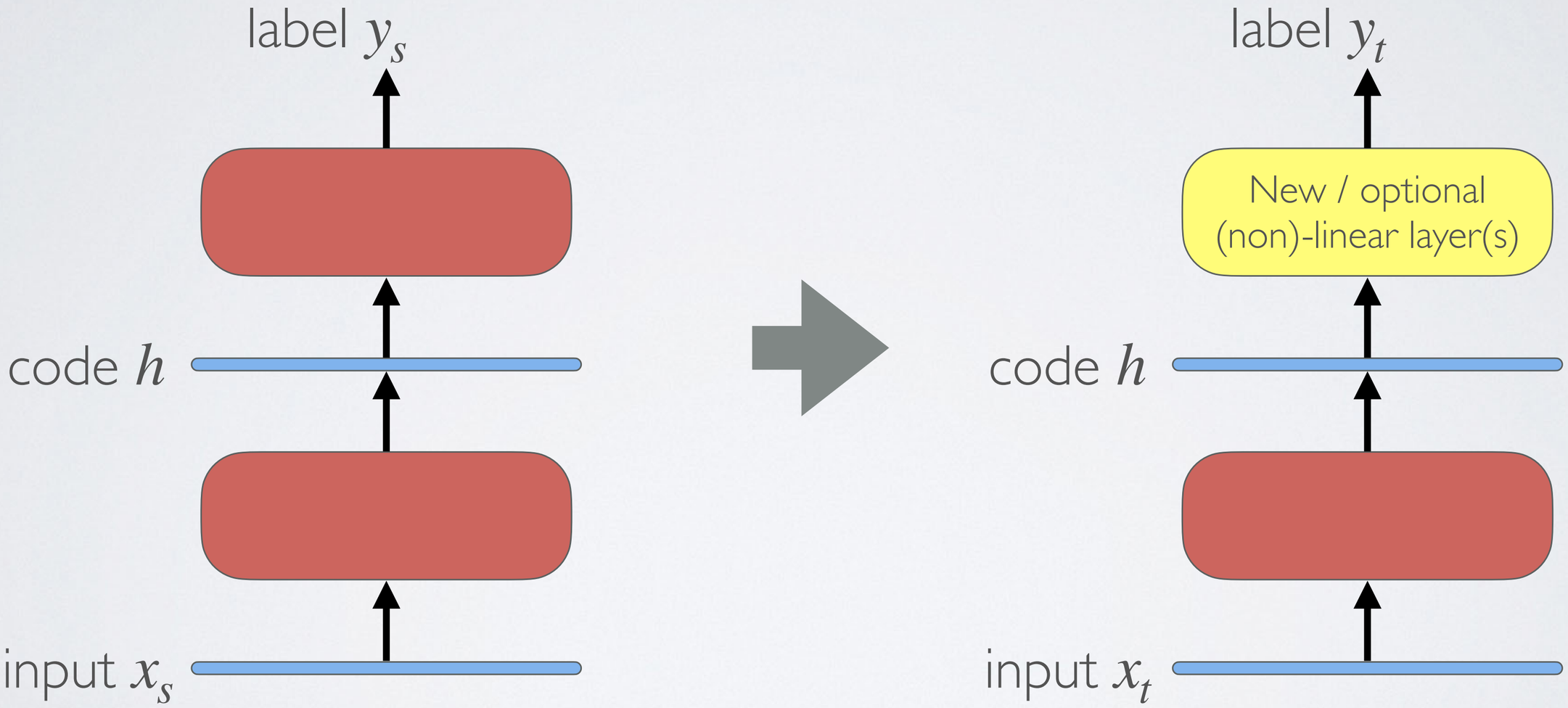
UNSUPERVISED / SELF-SUPERVISED LEARNING AS A REGULARIZATION STRATEGY





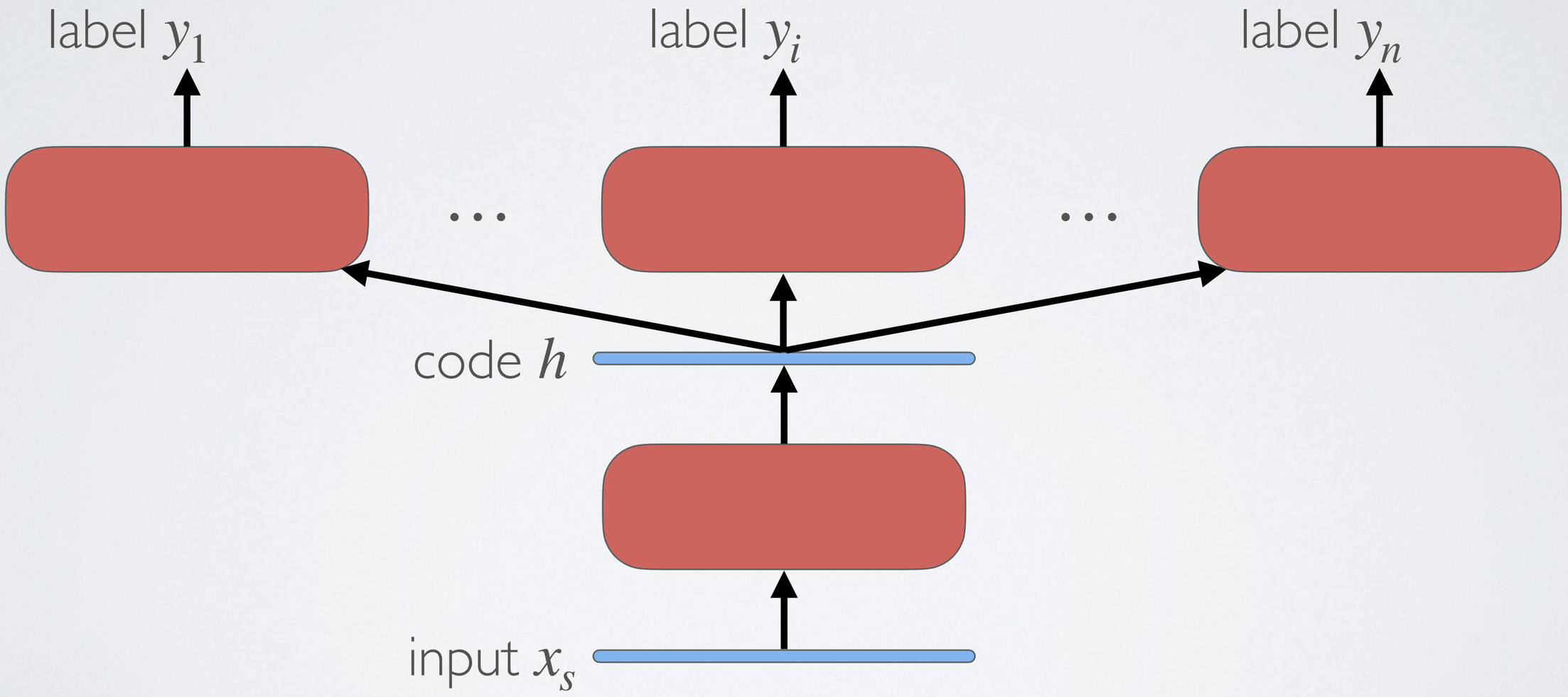
TRANSFER LEARNING AS A REGULARIZATION STRATEGY

Idea: pretrain some or all of the discriminative model parameters on another (related) source task: $P(y_s \mid x_s)$ before **transferring** to the target task $P(y_t \mid x_t)$



MULTITASK LEARNING AS A REGULARIZATION STRATEGY

Idea: pretrain some of the discriminative model parameters on multiple (related) source tasks: $P(y_1 \mid x_1), P(y_2 \mid x_2) \ \ldots \ P(y_n \mid x_n)$. We (may) care about performance on all tasks.



LABEL SMOOTHING

- **Label smoothing**: regularization technique that introduces noise for the labels
- Assume a K-class classification problem, and that the dataset label y (one-hot vector of dim. K) is correct with probability $1-\alpha$ (for small α).
- Label smoothing replaces the hard 0 and 1 classification targets with $\frac{\alpha}{K-1}$ and $1-\alpha$ respectively, otherwise use cross-entropy loss (CE) as usual.
- Eg., $K=3$ and $\alpha=0.01$: $y=\begin{bmatrix}0\\1\\0\end{bmatrix}$, $y_{\text{ls}}=\begin{bmatrix}0.005\\0.990\\0.005\end{bmatrix}$
- **Effect**: Clusters representations of same class in the penultimate layer of the network.

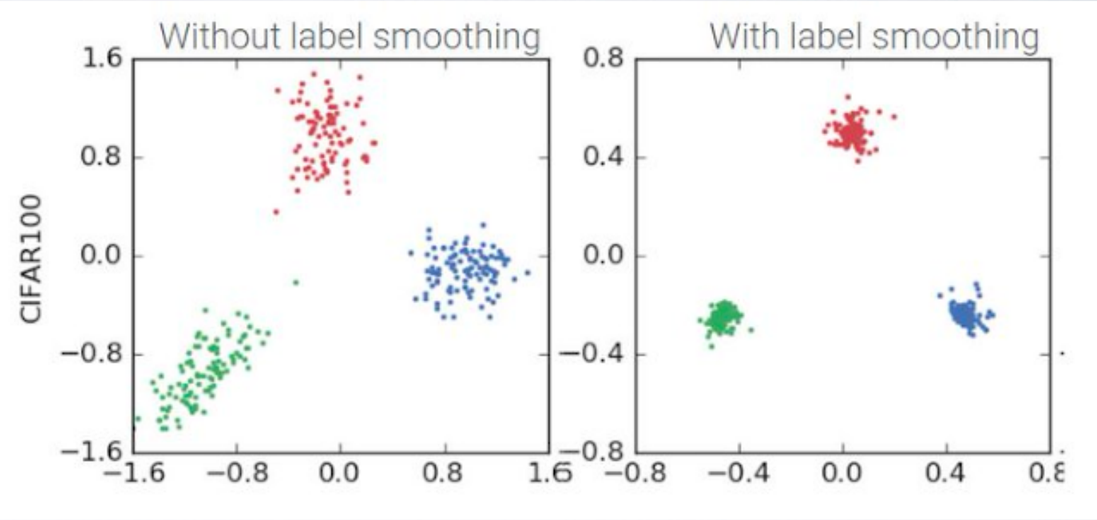
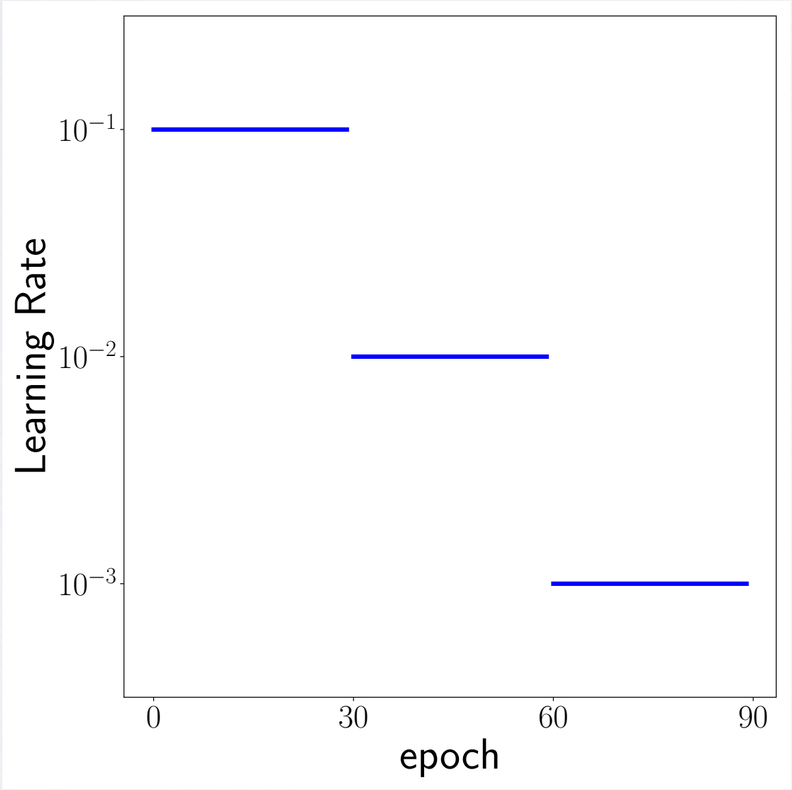


Image Source: When Does Label Smoothing Help?

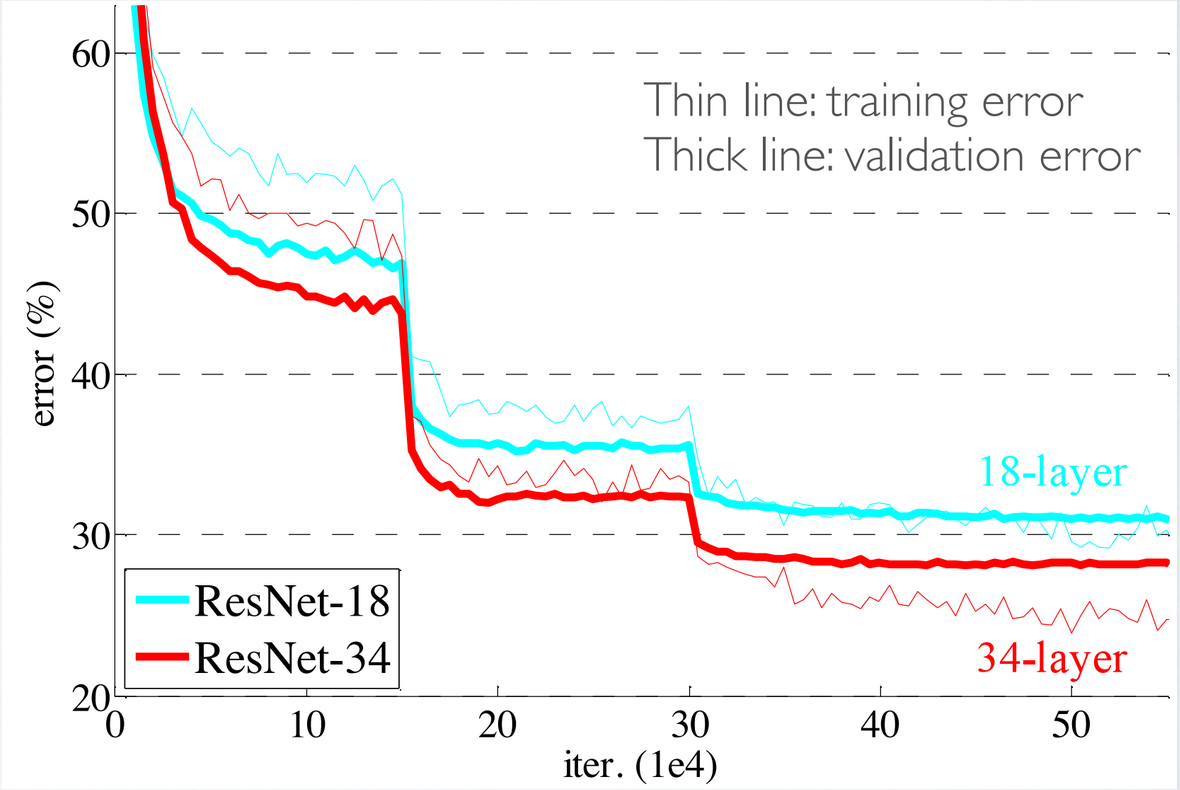
Learning Rate Decay

DISCRETE STAIRCASE LEARNING RATE DECAY

- Continuous decreasing learning rates are used (linear, exponential decay, 1/t decay)
- Recently popular: a fixed schedule of decreasing discrete learning rates.



(a) Learning rate decay strategy



(b) Figure taken from He et al. (2016)

Image source
You et al. (2019)

DISCRETE STAIRCASE LEARNING RATE DECAY

Large initial learning rates:

- accelerate learning,

- improve generalization by avoiding early overfitting.

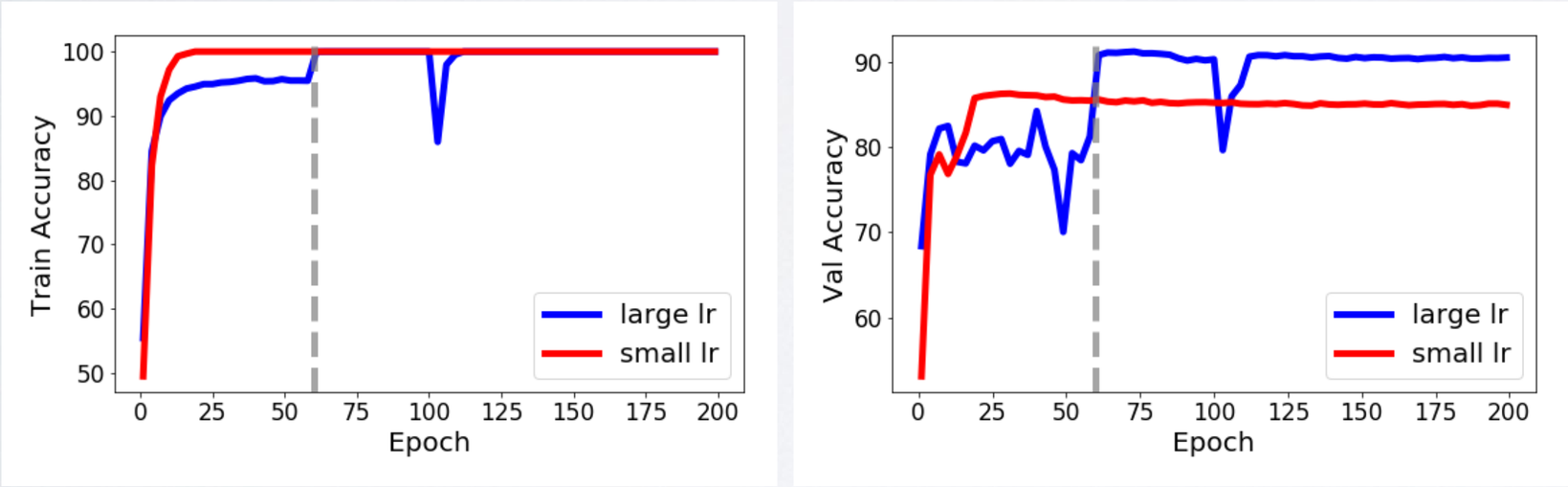


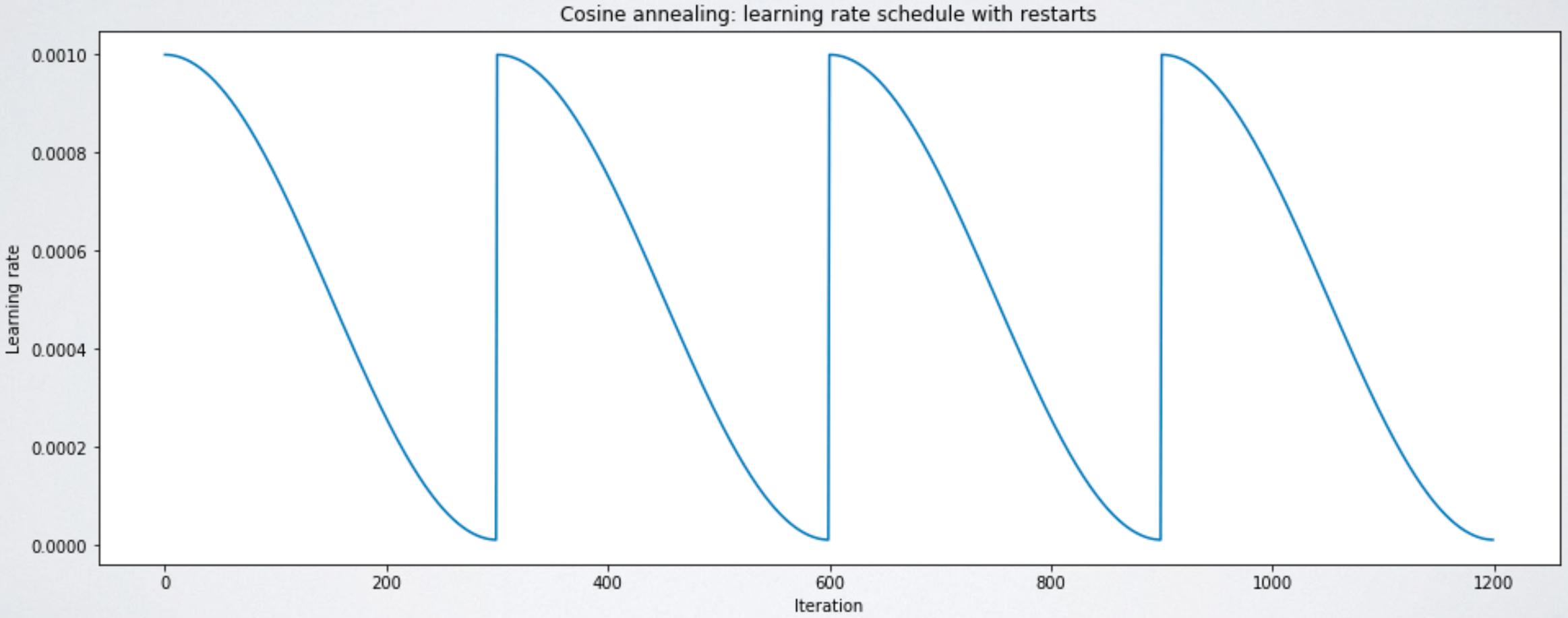
Figure 1: CIFAR-10 accuracy vs. epoch for WideResNet with weight decay, no data augmentation, and initial lr of 0.1 vs. 0.01. Gray represents the annealing time. **Left:** Train. **Right:** Validation.

Image source: Li et al (2020), *Towards Explaining the Regularization Effect of Initial Large Learning Rate in Training Neural Networks*

COSINE LEARNING RATE DECAY WITH RESTARTS

$$\eta_t = \eta^i_{\min} + \frac{1}{2}\left(\eta^i_{\max} - \eta^i_{\min}\right)\left(1 + \cos\left(\frac{T_{\text{current}}}{T_i}\pi\right)\right)$$

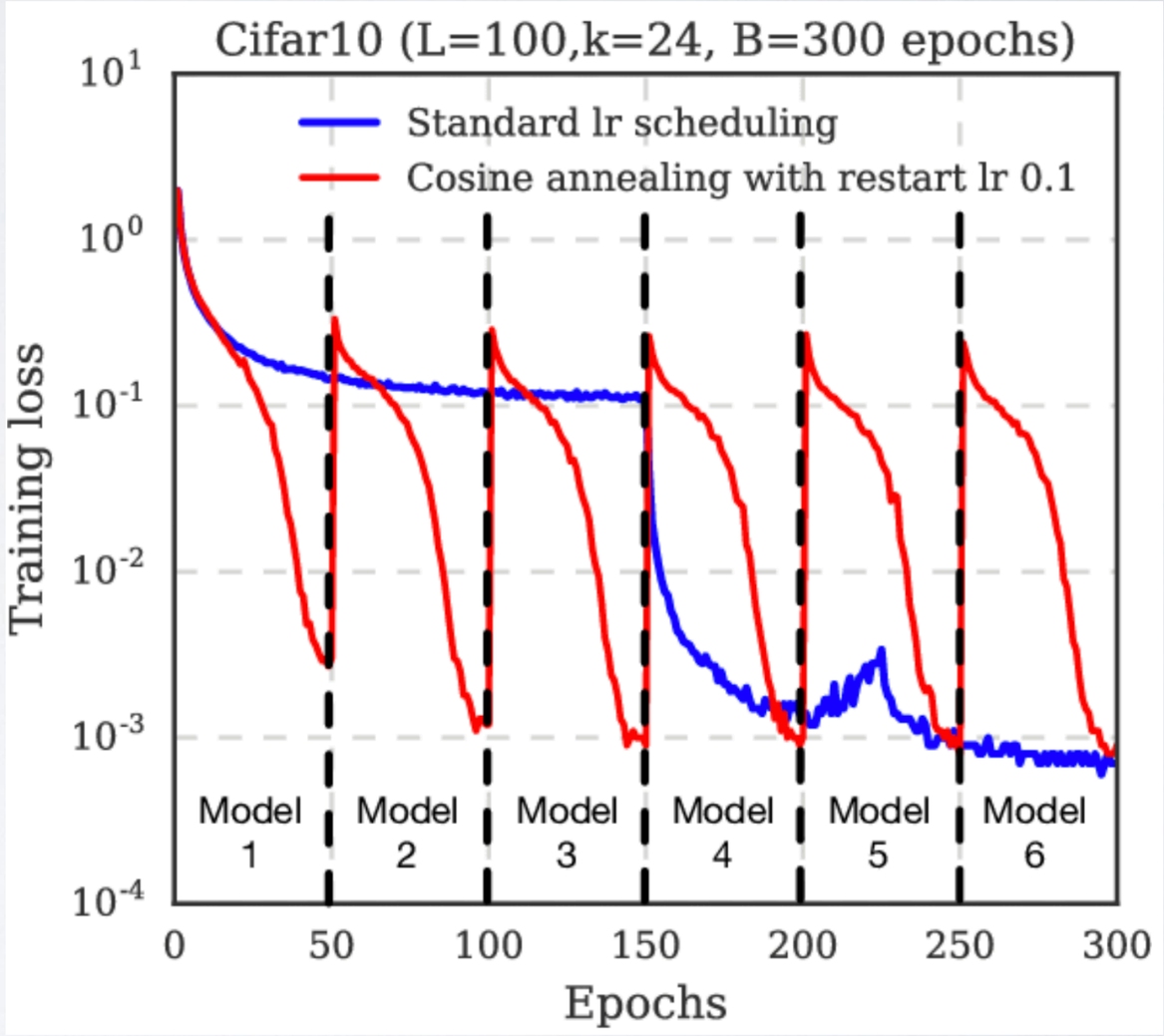
$\eta^i_{\min}$ - minimum learning rate
$\eta^i_{\max}$ - maximum learning rate
T_{current} - number of epochs since last restart
T_i - number of epochs in an cycle



COSINE LEARNING RATE DECAY WITH RESTARTS

$$\eta_t = \eta_{\min}^i + \frac{1}{2}\left(\eta_{\max}^i - \eta_{\min}^i\right)\left(1 + \cos\left(\frac{T_{\text{current}}}{T_i}\pi\right)\right)$$

- Restarts allow the model to leave solutions that generalize poorly.
- Similar effect to large initial learning rates.



DATA AUGMENTATION: RANDAUGMENT

Cubuk et al (2019) *RandAugment: Practical automated data augmentation with a reduced search space*

- RandAugment is among the most popular augmentation strategies: it is both effective and easy to use.
- All transformation are equally probable.

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

   Args:
     N: Number of augmentation transformations to
          apply sequentially.
     M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

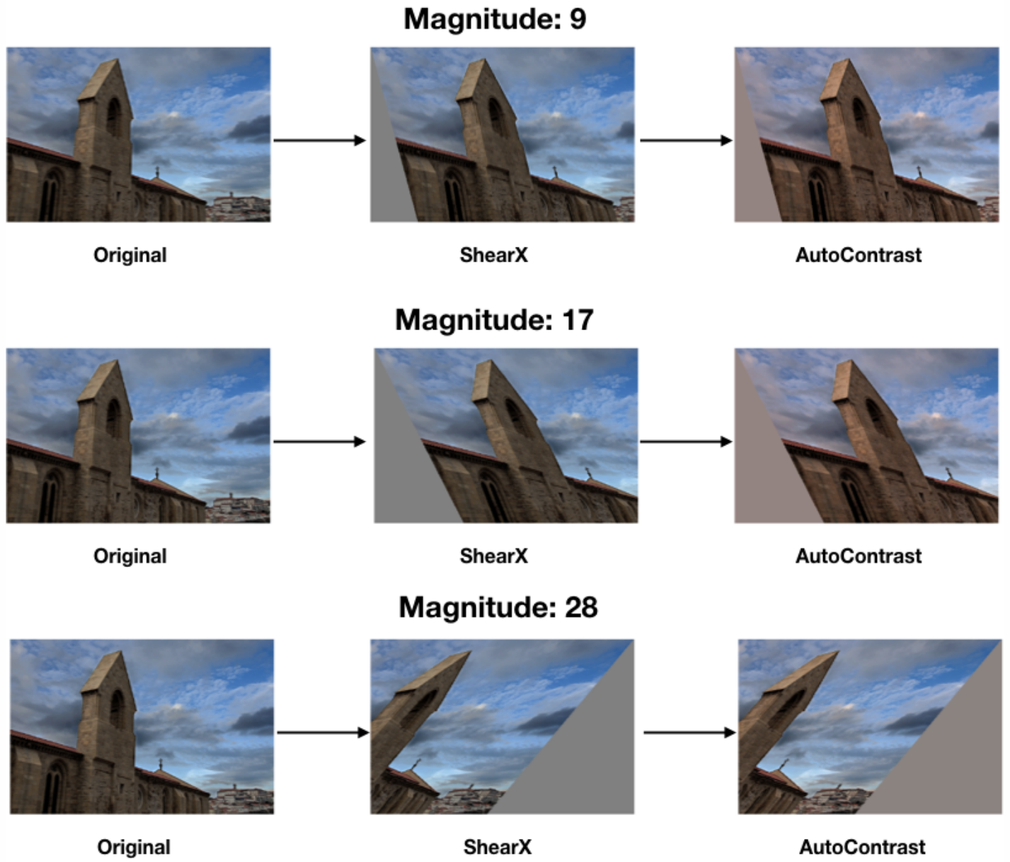


Figure 1. **Example images augmented by RandAugment.** In these examples N=2 and three magnitudes are shown corresponding to the optimal distortion magnitudes for ResNet-50, EfficientNet-B5 and EfficientNet-B7, respectively. As the distortion magnitude increases, the strength of the augmentation increases.

DATA AUGMENTATION: RANDAUGMENT

Cubuk et al (2019) *RandAugment: Practical automated data augmentation with a reduced search space*

- RandAugment is among the most popular augmentation strategies: it is both effective and easy to use.
- All transformation are equally probable.

		search space	CIFAR-10 PyramidNet	SVHN WRN	ImageNet ResNet	ImageNet E. Net-B7
	Baseline	0	97.3	98.5	76.3	84.0
AutoAugment:	AA	10^{32}	98.5	98.9	77.6	84.4
Fast AutoAugment:	Fast AA	10^{32}	98.3	98.8	77.6	-
Population-Based Augmentation:	PBA	10^{61}	98.5	98.9	-	-
RandAugment:	RA (ours)	10^{2}	98.5	99.0	77.6	85.0

DATA AUGMENTATION: RANDAUGMENT

Cubuk et al (2019) *RandAugment: Practical automated data augmentation with a reduced search space*

Equalize: Equalize the image histogram.

Source: https://tugot17.github.io/data-science-blog/albumentations/data-augmentation/tutorial/2020/09/20/Pixel-level-transforms-using-albumentations-package.html

DATA AUGMENTATION: RANDAUGMENT

Cubuk et al (2019) *RandAugment: Practical automated data augmentation with a reduced search space*

Posterize: Reduce the number of bits for each color channel.

Source: https://tugot17.github.io/data-science-blog/albumentations/data-augmentation/tutorial/2020/09/20/Pixel-level-transforms-using-albumentations-package.html

DATA AUGMENTATION: RANDAUGMENT

Cubuk et al (2019) *RandAugment: Practical automated data augmentation with a reduced search space*

Solarize: Invert all pixel values above a threshold.

Source: https://tugot17.github.io/data-science-blog/albumentations/data-augmentation/tutorial/2020/09/20/Pixel-level-transforms-using-albumentations-package.html

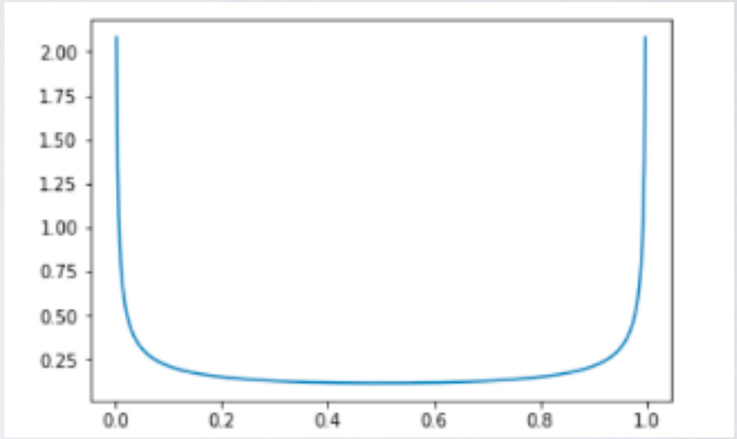
MIXUP

Zhang et al. (ICLR 2018) *mixup: BEYOND EMPIRICAL RISK MINIMIZATION*

- Train with a superposition of images and corresponding mix of target labels

$$\text{Image}_{\text{new}} = \alpha \times \text{Image}_1 + (1 - \alpha) \times \text{Image}_2$$

$$y_{\text{new}} = \alpha \times y_1 + (1 - \alpha) \times y_2$$



$p(\alpha) = \text{beta distribution}$

Image source: https://towardsdatascience.com/2-reasons-to-use-mixup-when-training-yor-deep-learning-models-58728f15c559

RANDOM ERASING / CUTOUT

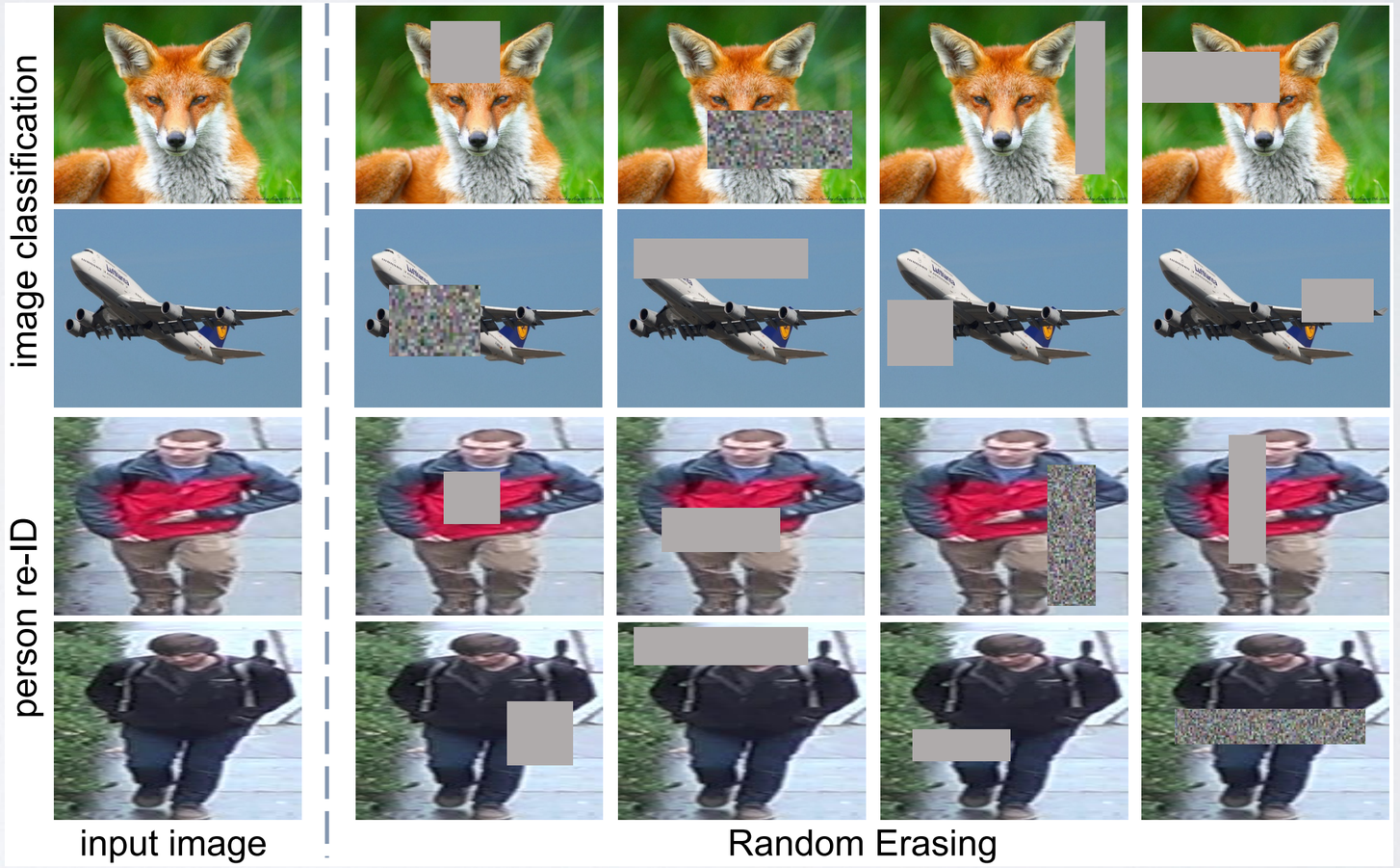
Zhong et al (2017) *Random Erasing Data Augmentation*
DeVries & Taylor (2017) *Improved Regularization of Convolutional Neural Networks with Cutout*

<u>Random erasing training procedure:</u>

Image randomly undergoes either of the two operations:

1) kept unchanged;

2) randomly choose a rectangle region (arbitrary size), assign pixels within the selected region with random values (or the dataset mean pixel value).



CUTMIX

Yun et al. (2019) *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*

$$\text{Image}_{\text{new}} = \mathbf{M} \odot \text{Image}_1 + (\mathbf{1} - \mathbf{M}) \odot \text{Image}_2 \quad \text{where } \mathbf{M} \text{ has bounding box } B = (r_x, r_y, r_w, r_h)$$

$$y_{\text{new}} = \lambda \times y_1 + (1 - \lambda) \times y_2$$

$$r_x \sim \text{Unif}(0, W), \quad r_w = W\sqrt{1-\lambda}$$

$$r_y \sim \text{Unif}(0, H), \quad r_h = H\sqrt{1-\lambda}$$

- CutMix combines Mixup label mixing and CutOut-like image patches,

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4
ImageNet Cls (%)	76.3 (+0.0)	77.4 (+1.1)	77.1 (+0.8)	**78.6** **(+2.3)**
ImageNet Loc (%)	46.3 (+0.0)	45.8 (-0.5)	46.7 (+0.4)	**47.3** **(+1.0)**
Pascal VOC Det (mAP)	75.6 (+0.0)	73.9 (-1.7)	75.1 (-0.5)	**76.7** **(+1.1)**

STOCHASTIC DEPTH

Huang et al (ECCV 2016)
Deep Networks with Stochastic Depth

- $l =$ ResBlock index,
- $b_l \in \{0,1\}$ is Bernoulli rand. var. and a binary mask for ResBlock l.
- $p_l = \Pr(b_l = 1)$, for ResBlock l.
- We linearly decay p_l :

$$p_l = 1 - \frac{l}{L}(1 - p_L) .$$

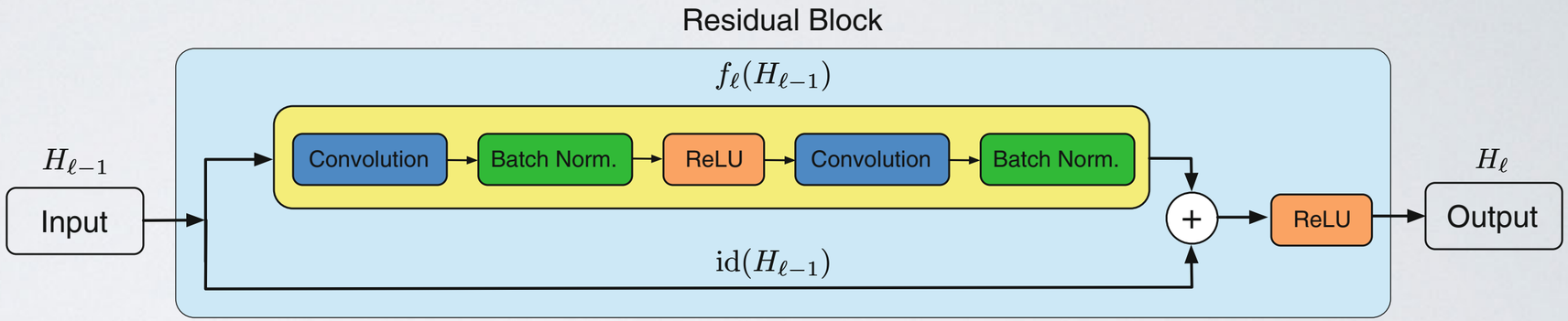
- Stochastic Depth Training:

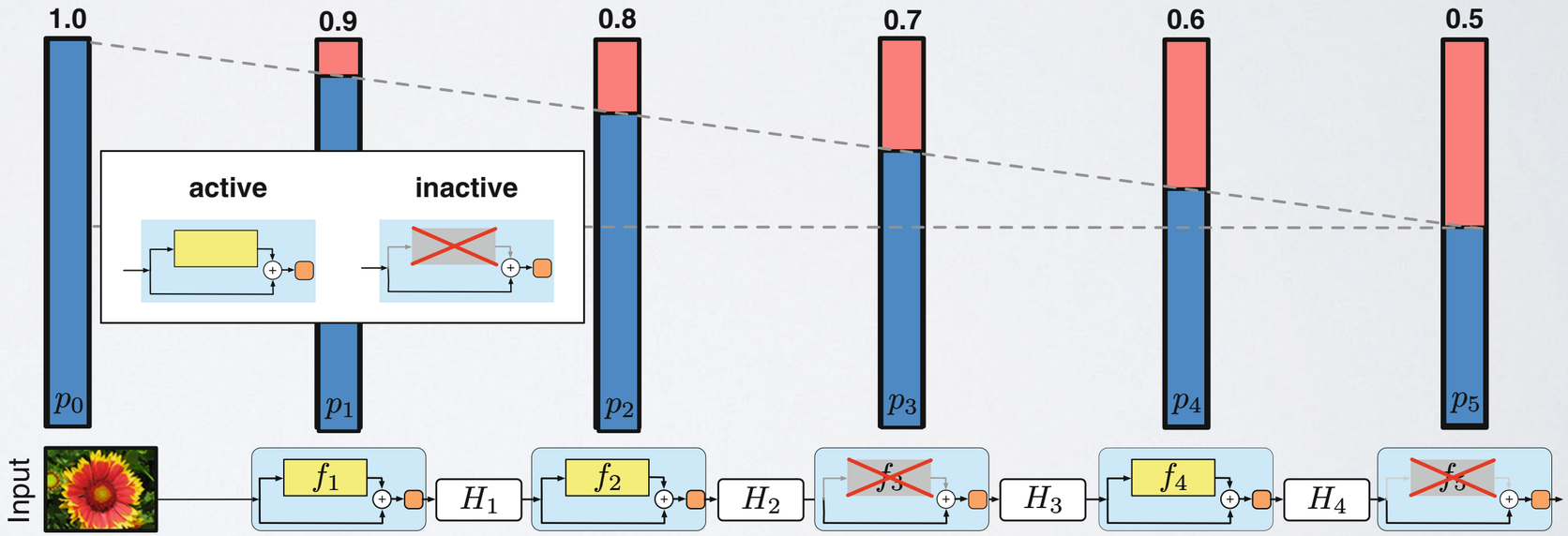
$$H_l = \text{ReLU}\left(b_l f_l(H_l - 1) + \text{id}(H_l - 1)\right) .$$

- reformulation of the ResBlock.

- Stochastic Depth Test time prop:

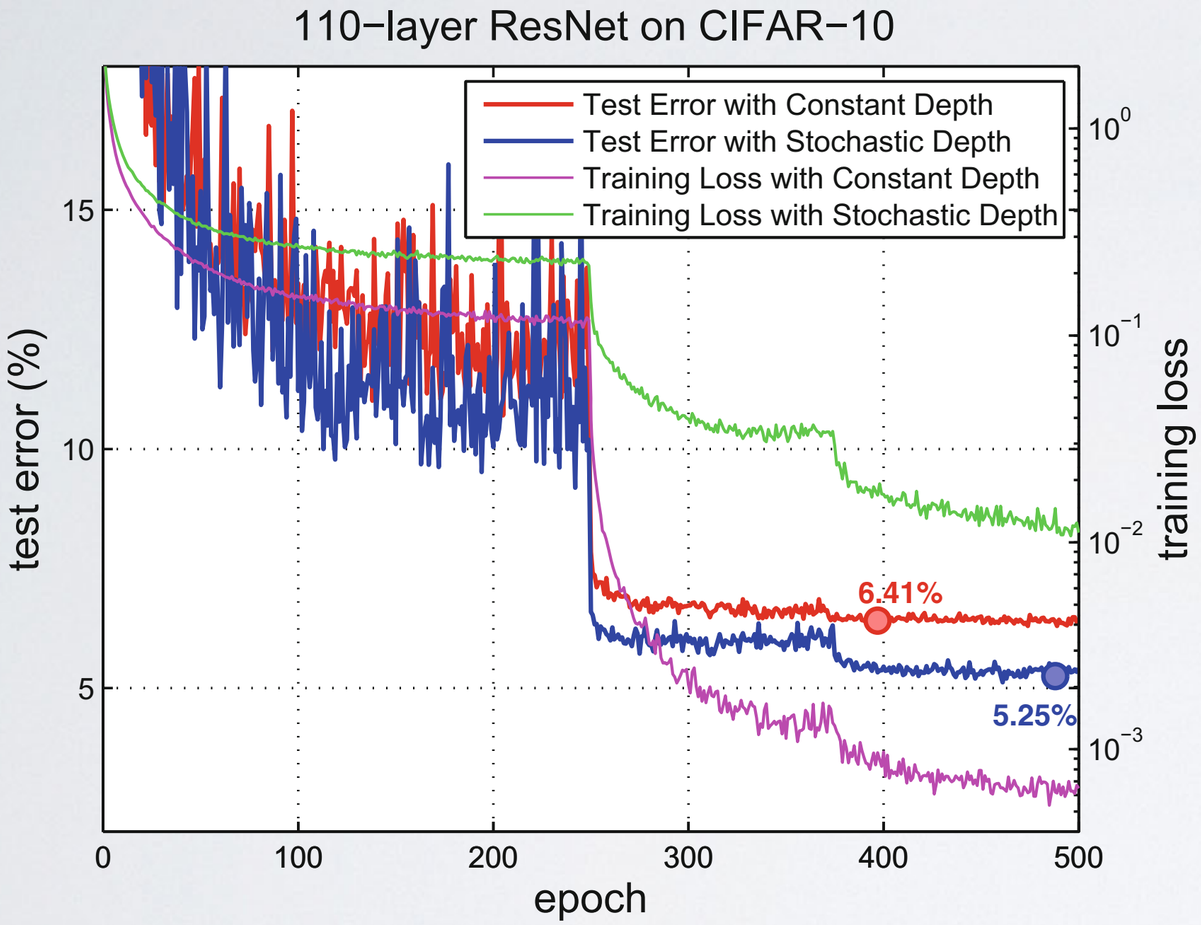
$$H_l^{\text{Test}} = \text{ReLU}\left(p_l f_l(H_{l-1}^{\text{Test}}) + H_{l-1}^{\text{Test}}\right) .$$

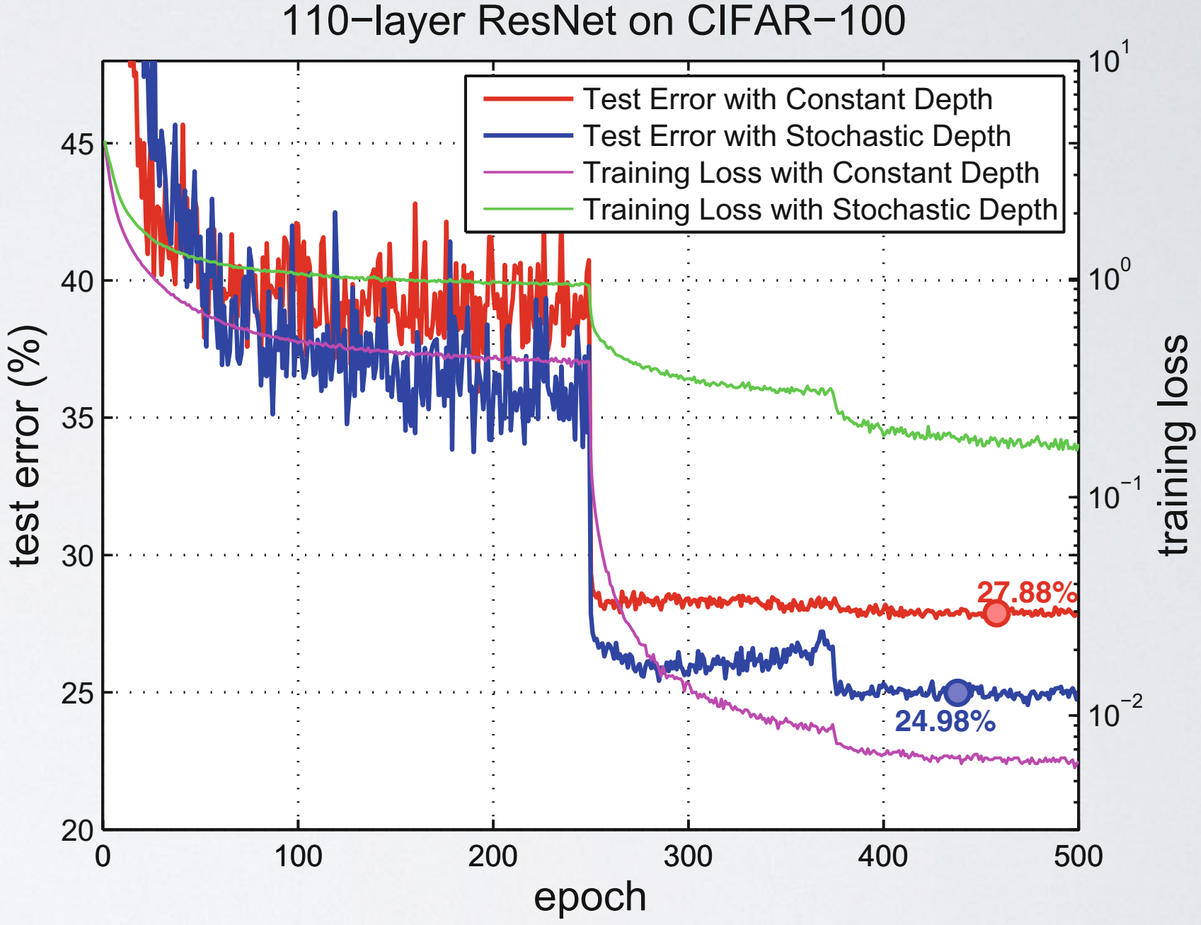




STOCHASTIC DEPTH

Huang et al (ECCV 2016)
Deep Networks with Stochastic Depth





STOCHASTIC DEPTH

Huang et al (ECCV 2016)
Deep Networks with Stochastic Depth

State of the art performance on CIFAR-10 with a 1202-layer Resnet!