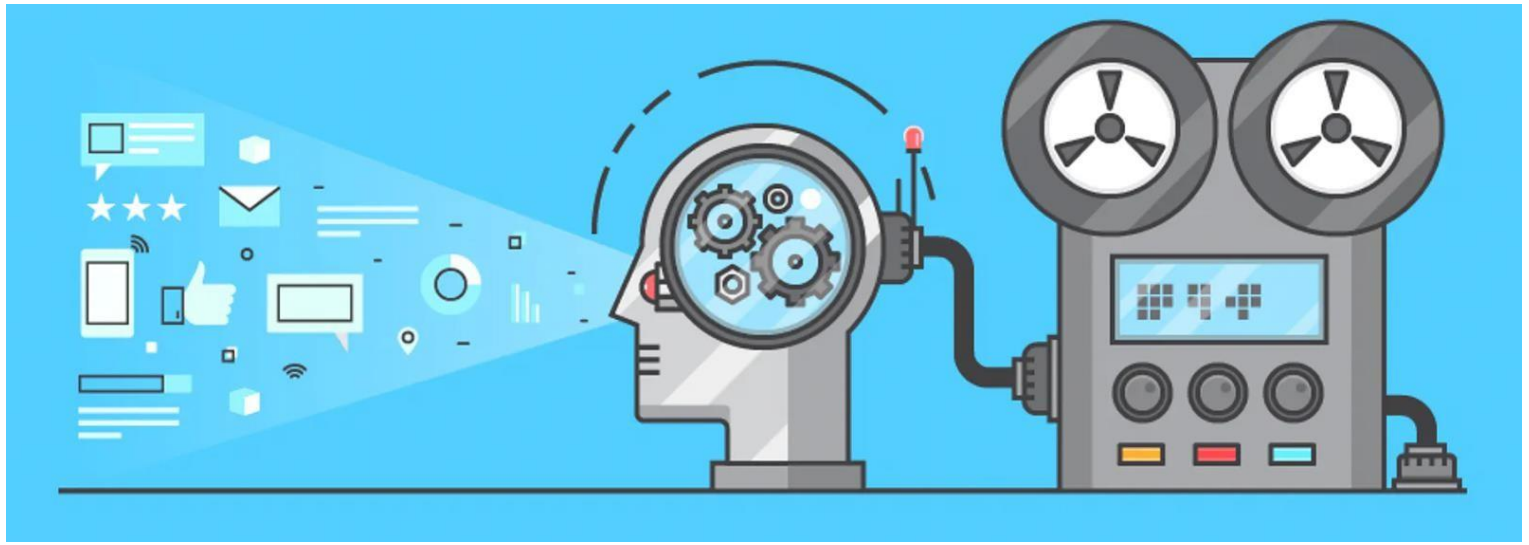
Deep Learning and Neural Networks

Topic 4:

Regularization



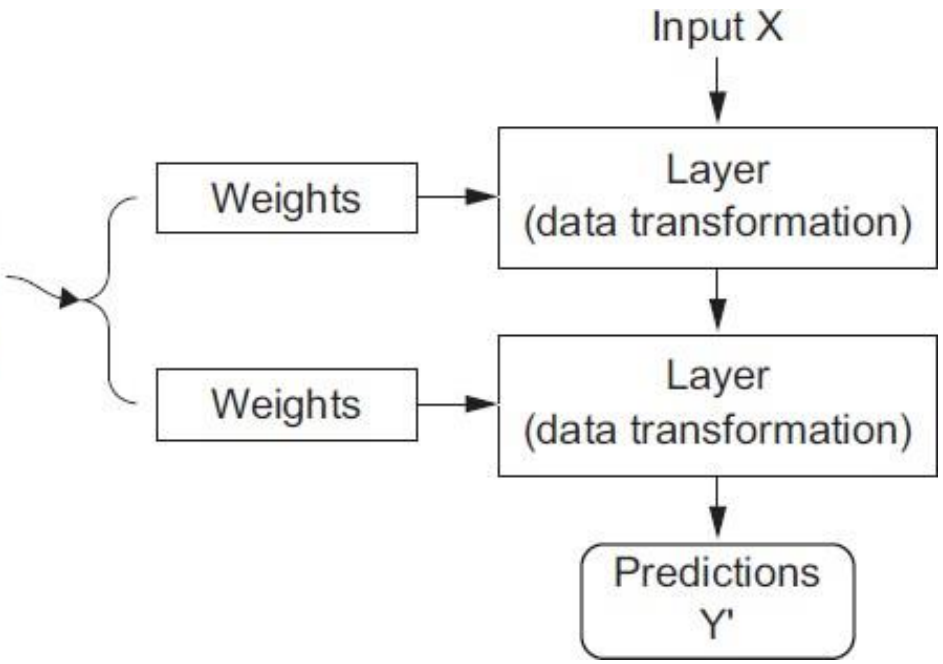
Ricardo Abel Espinosa Loera, McS

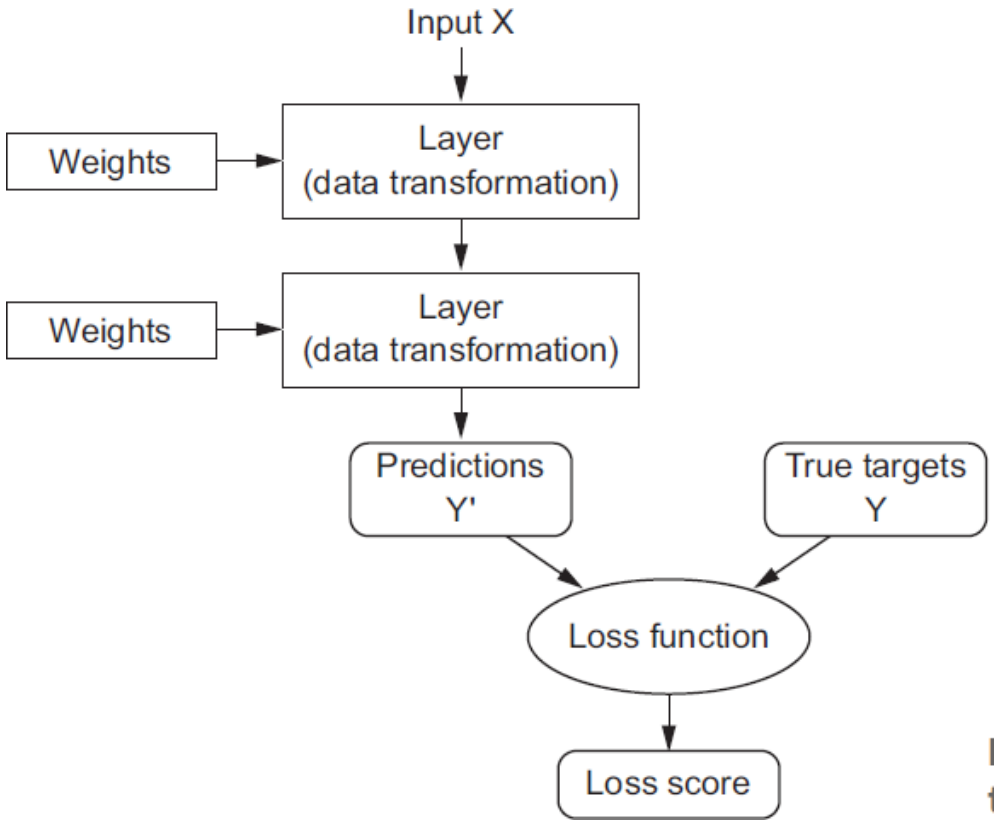
Researcher in DL & Computer Vision

Recap

Training neural networks

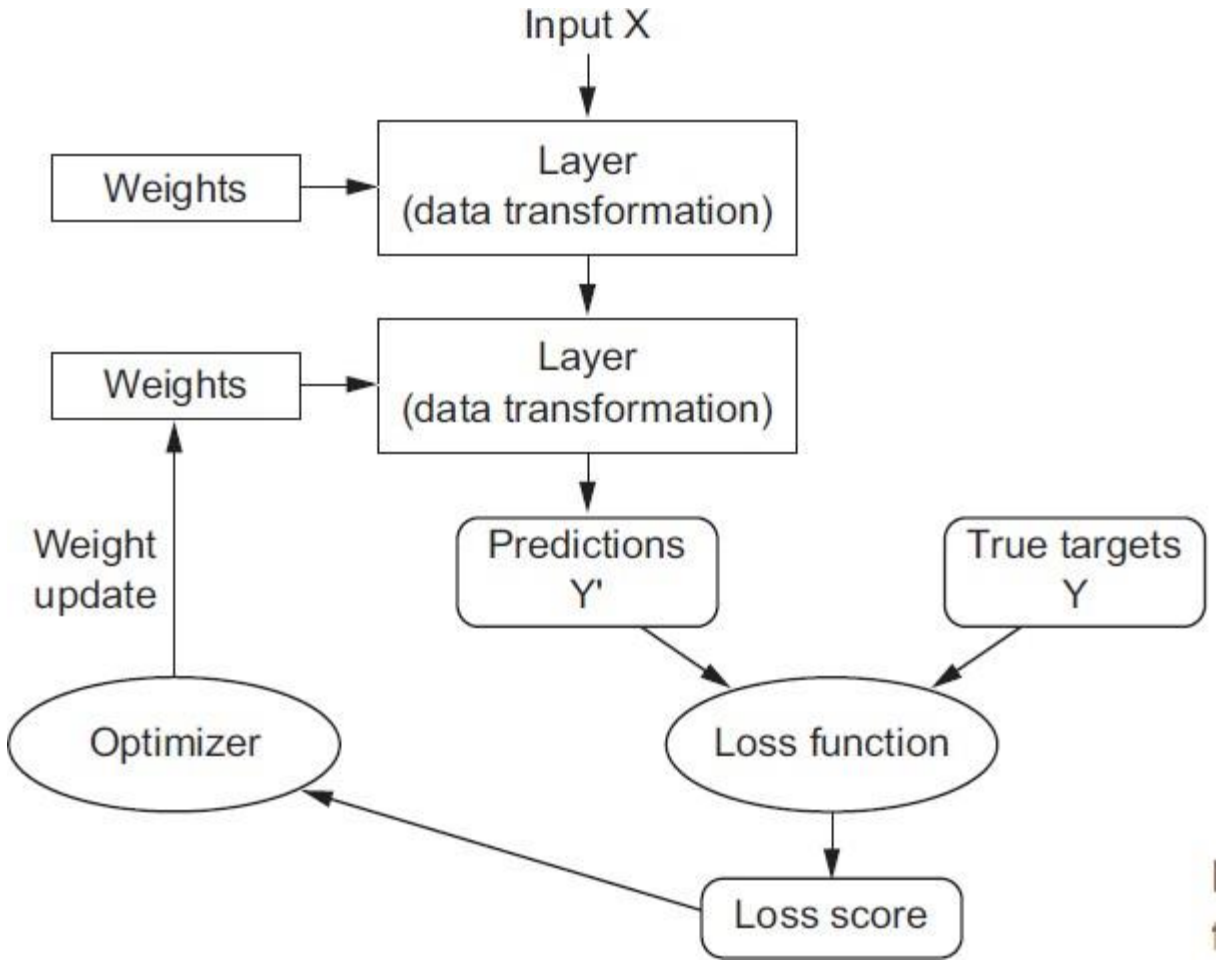
Goal: finding the right values for these weights





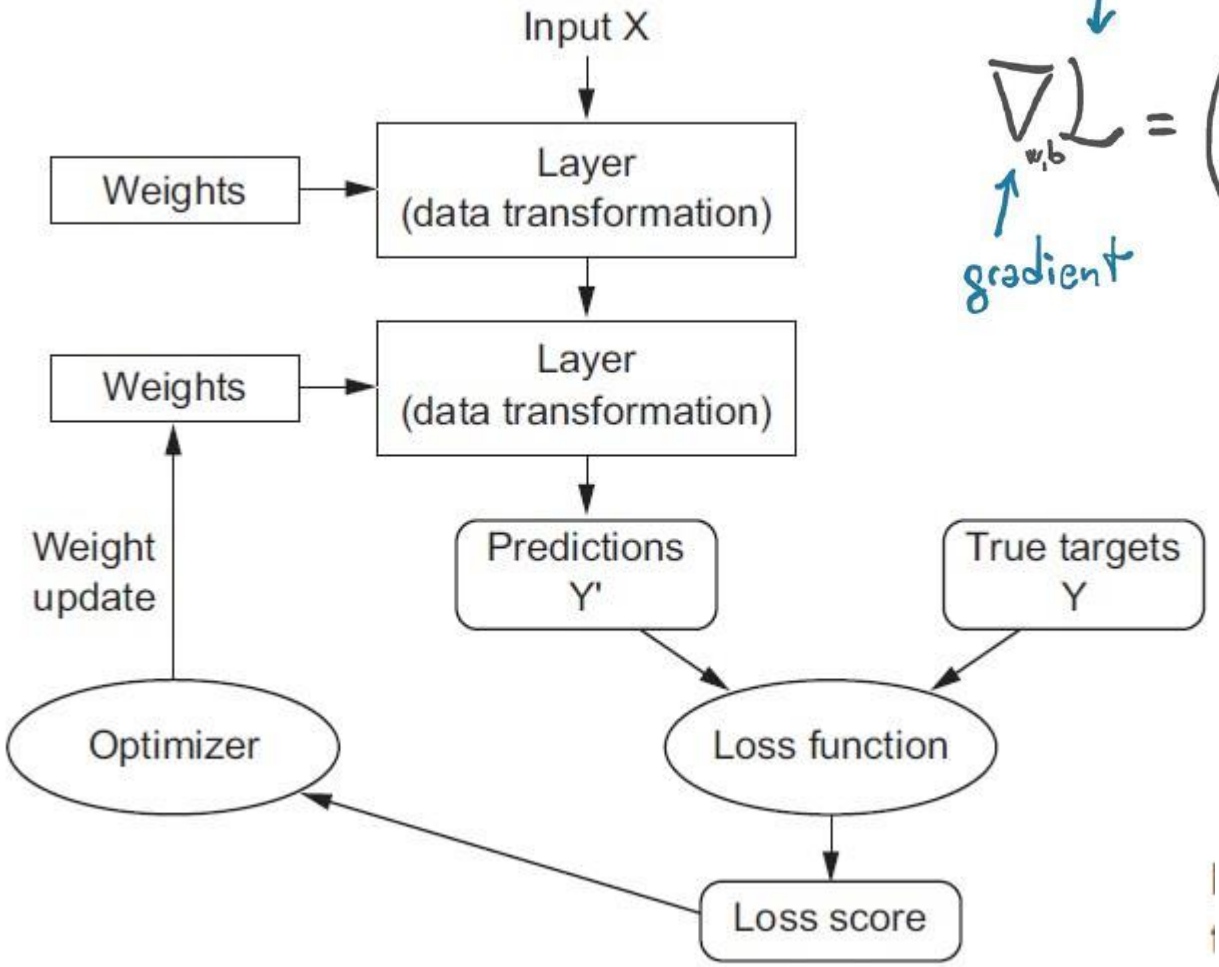
Recap

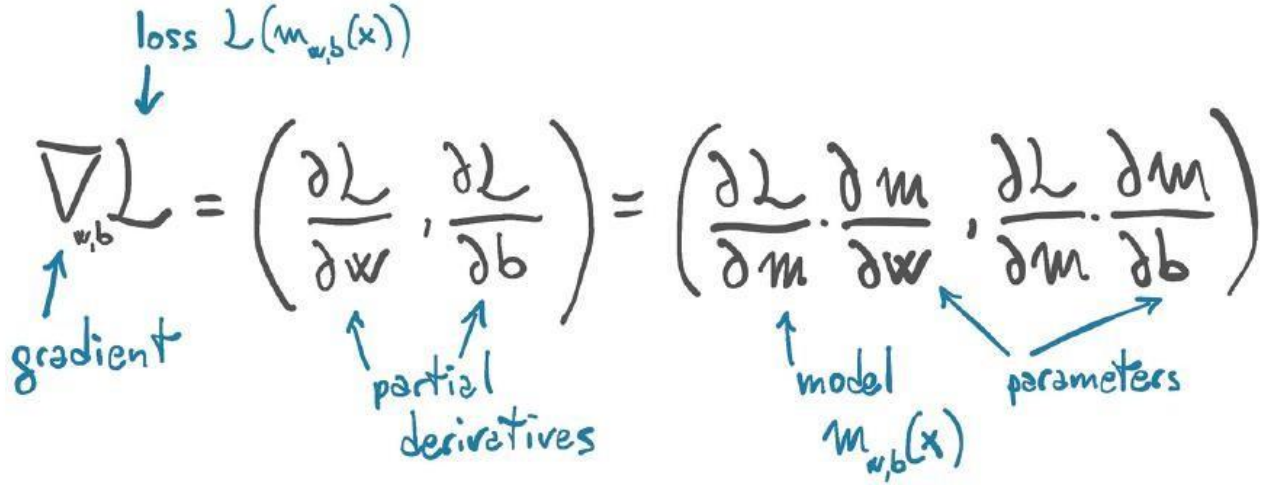
Training neural networks

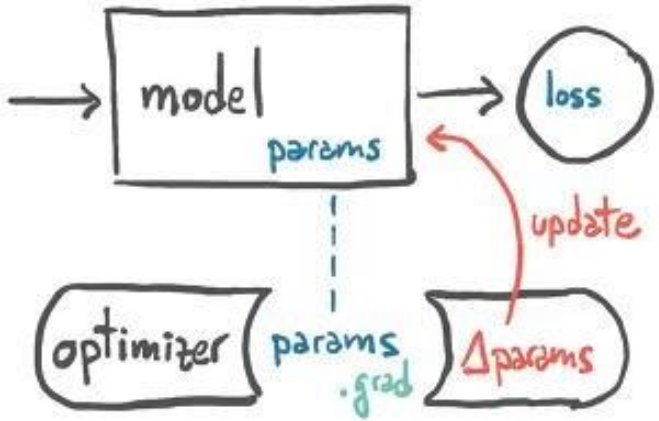


Recap

Training neural networks



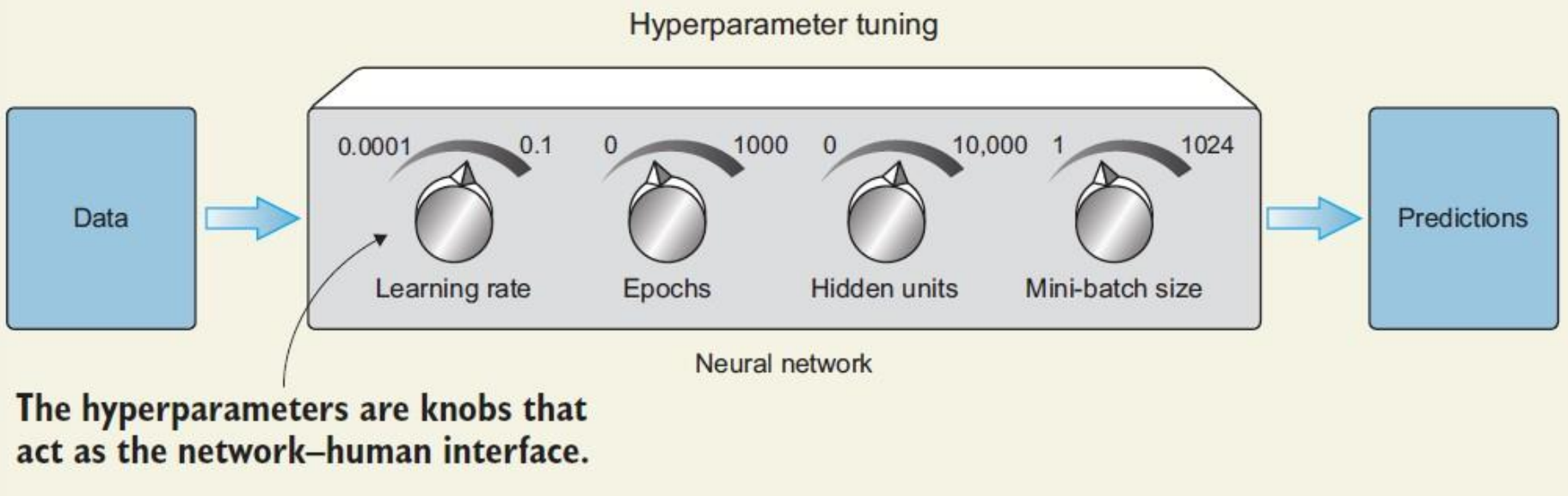




Recap
Training neural networks

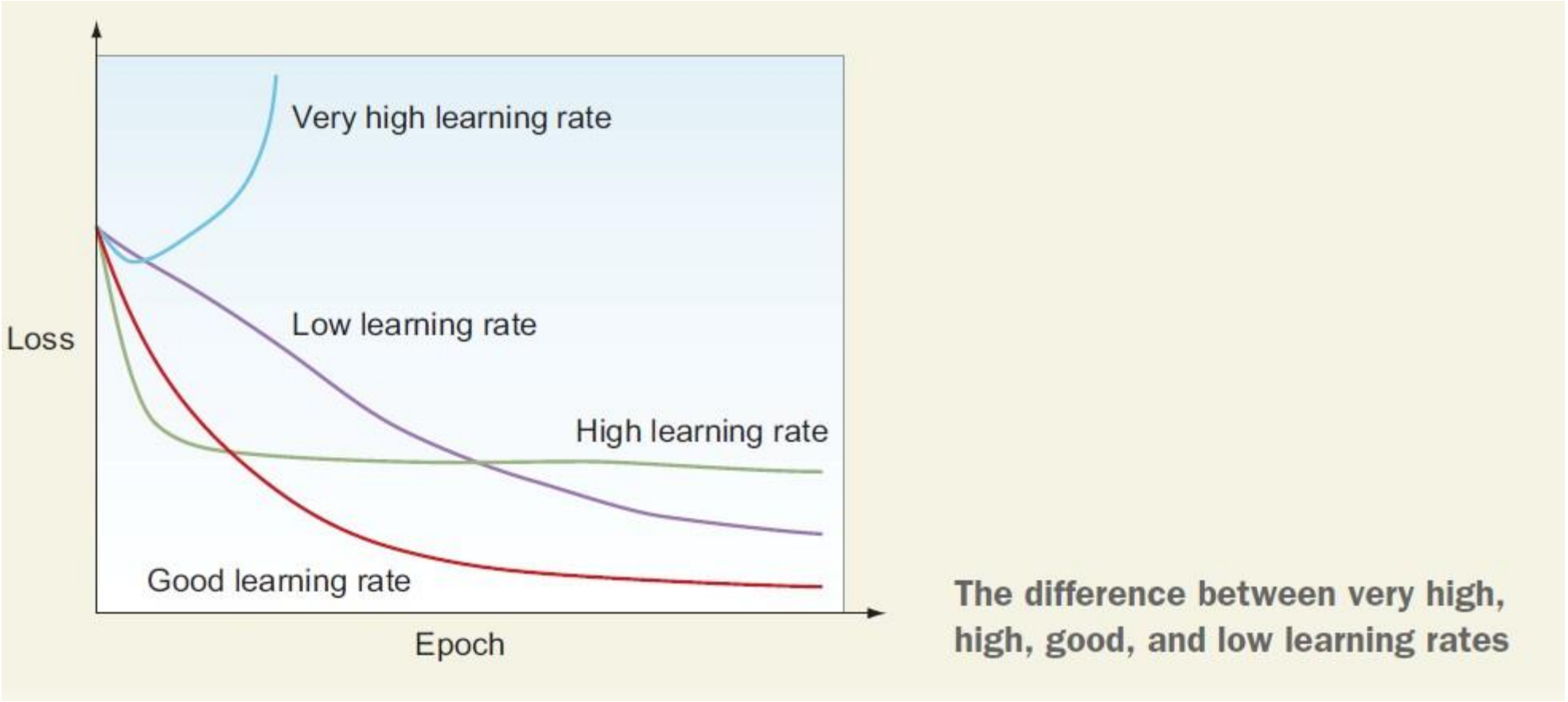
Turning the knobs

Think of hyperparameters as knobs on a closed box (the neural network). Our job is to set and tune the knobs to yield the best performance:



Recap

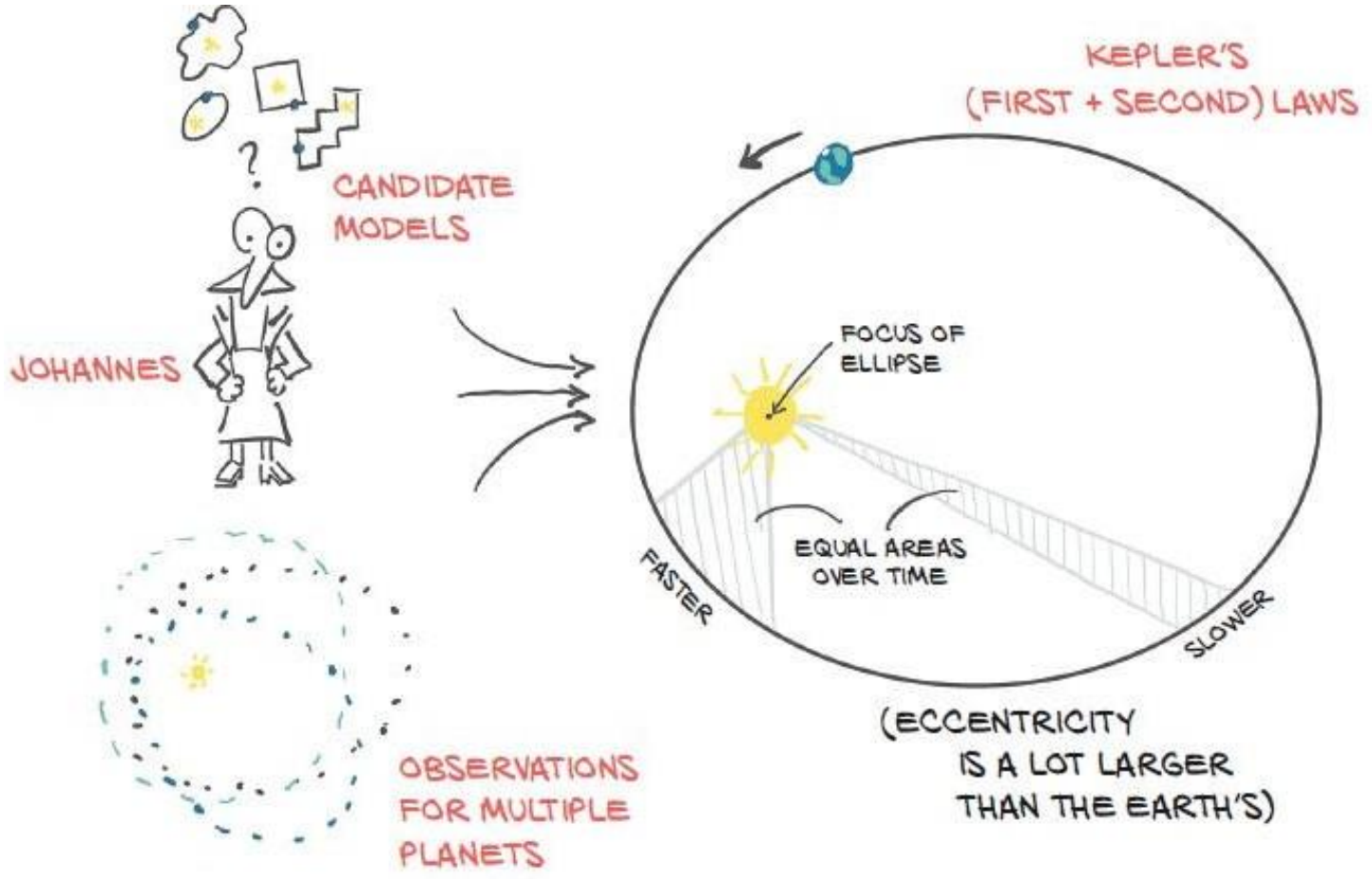
Training neural networks



The difference between very high, high, good, and low learning rates

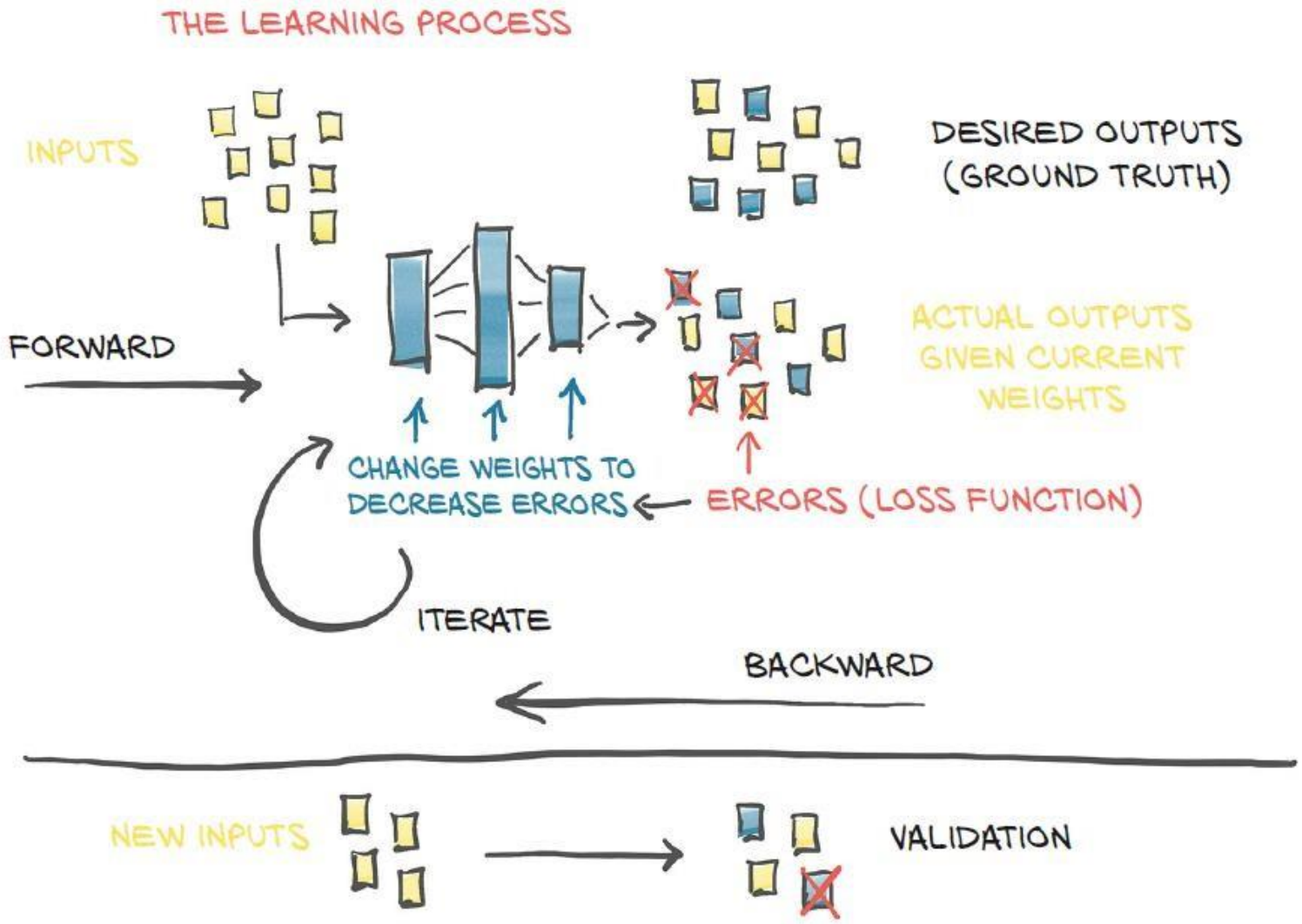
Recap

Training neural networks



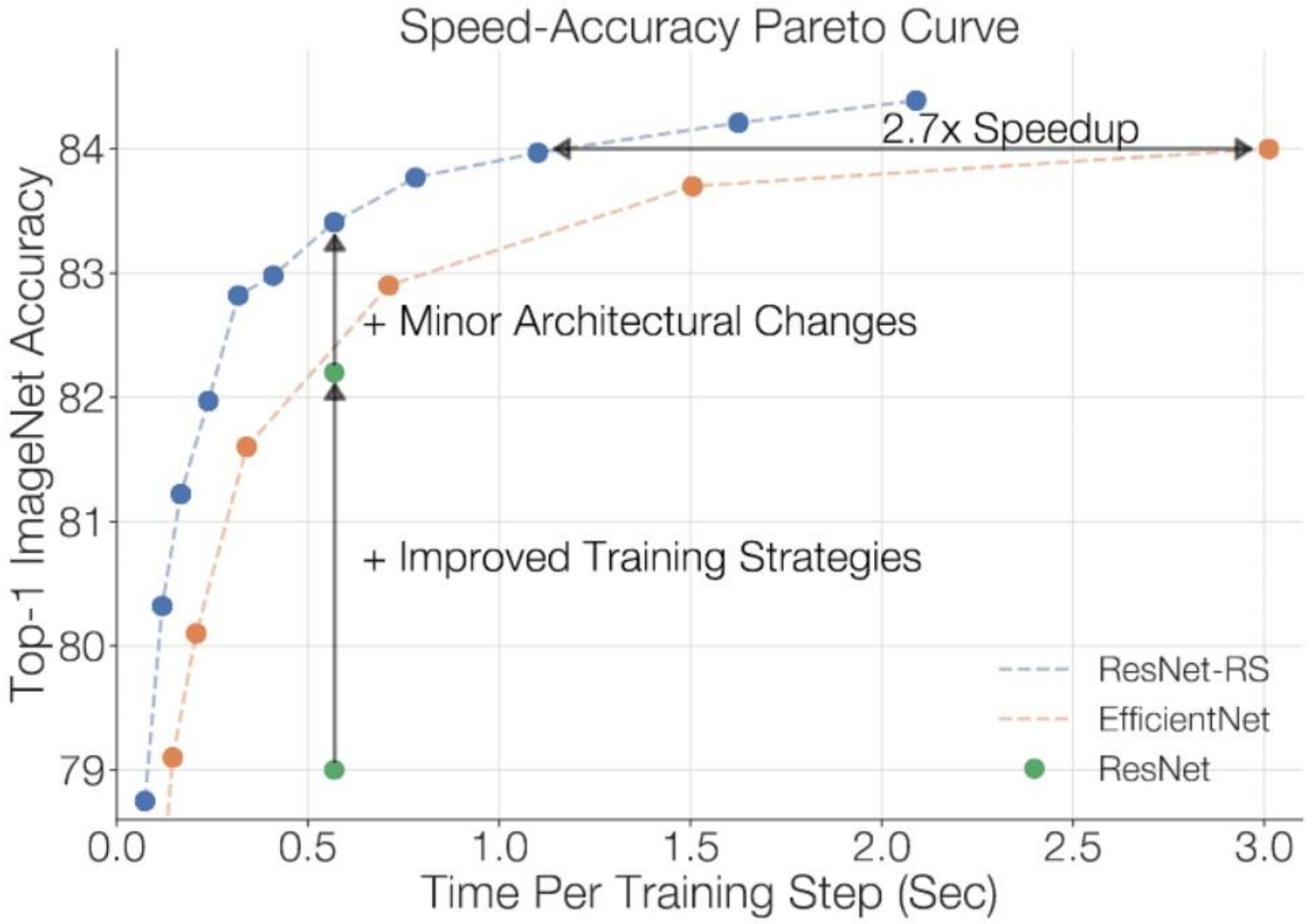
Recap

Training neural networks

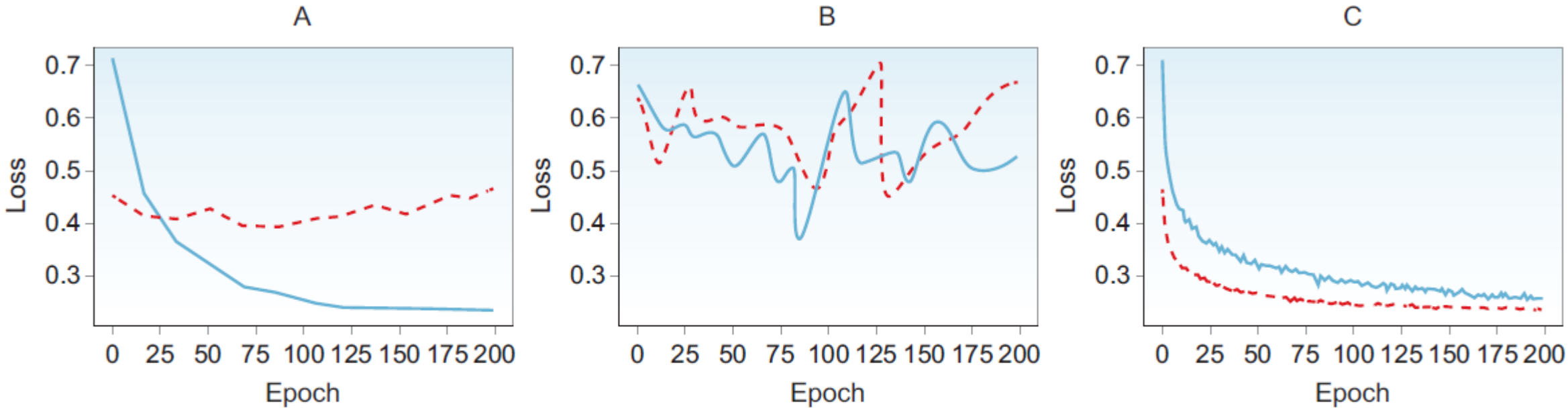


Training Neural Nets

Regularization



BIAS AND VARIANCE
in deep learning

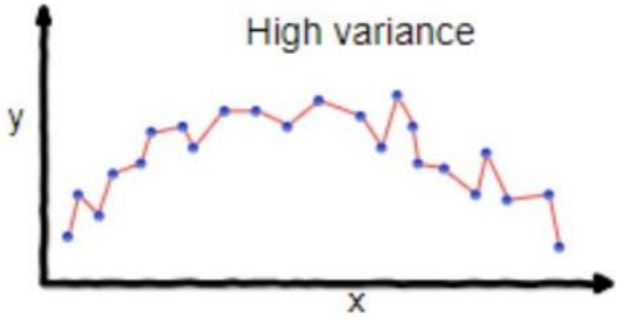


(A) The network improves the loss value on the training data but fails to generalize

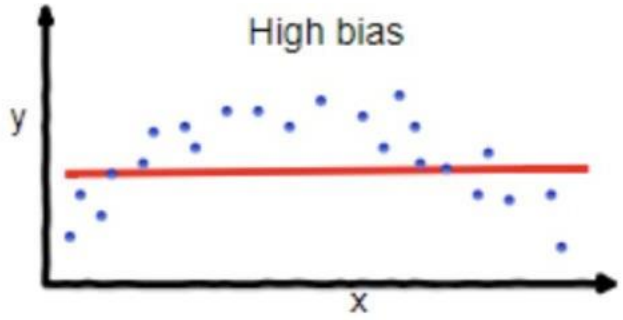
(B) The network performs poorly on both the training and validation data.

(C) The network learns the training data and generalizes to the validation data.

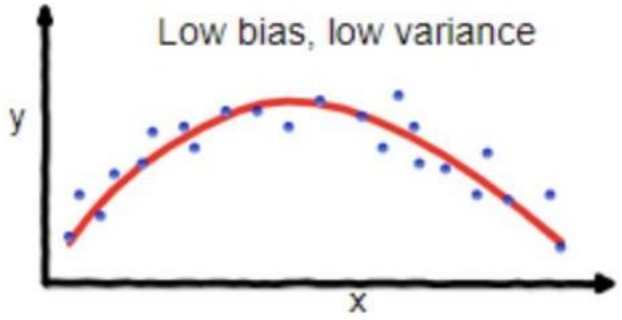
BIAS AND VARIANCE
in deep learning



overfitting



underfitting



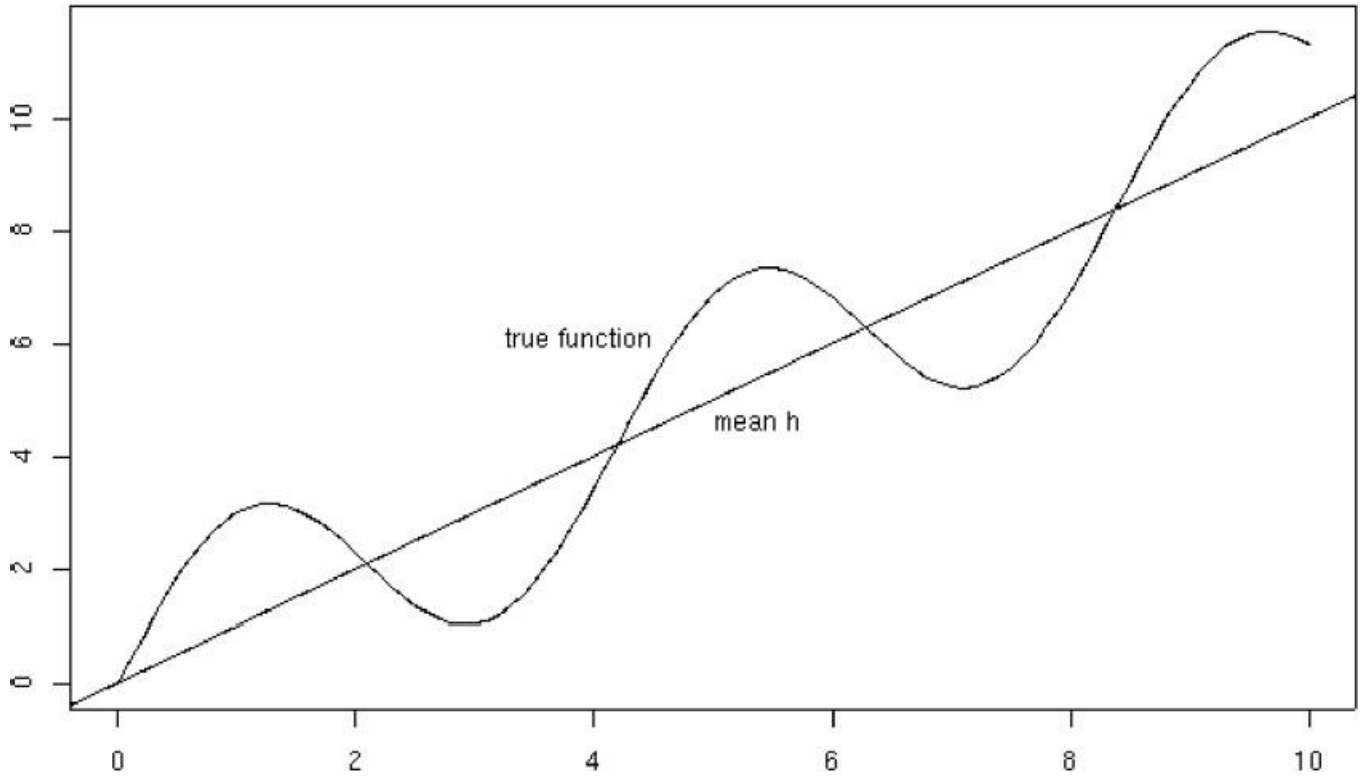
Good balance

BIAS AND VARIANCE
in deep learning

Bias

Discrepancy between averaged estimated and true function

$$Bias[h(x)] = \overline{h(x)} - f(x)$$



BIAS AND VARIANCE

Source of Bias

Inability to represent certain decision boundaries

- e.g.) linear classifier, decision trees.

Incorrect assumptions

- e.g.) linear assumption for data generated from a higher degree polynomial model.

Models that are too global (or too smooth)

- e.g.) a single linear separator

If the bias is high, the model is underfitting the data.

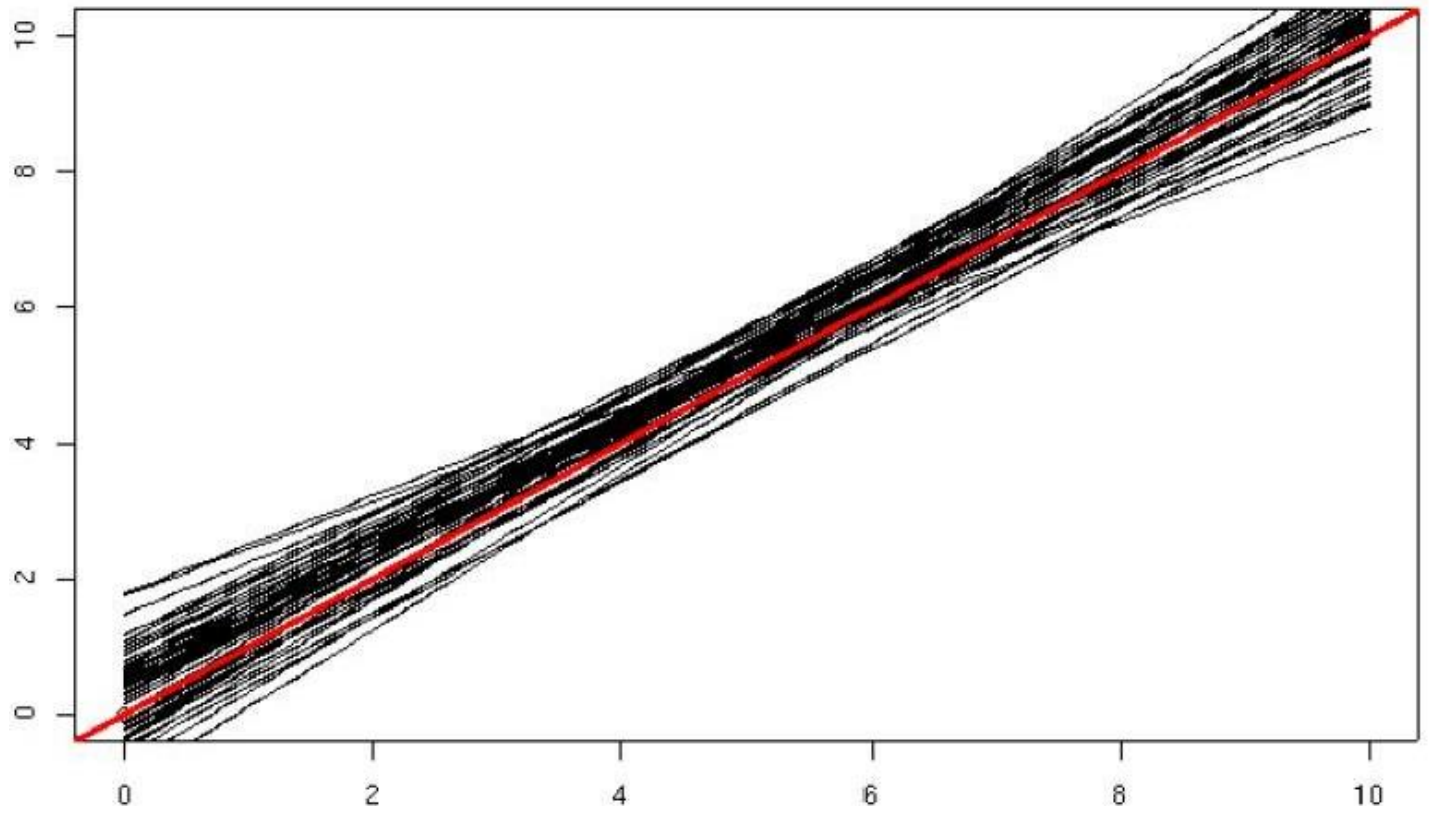
BIAS AND VARIANCE

Source of Bias

Variance

Discrepancy between models trained on different training sets

$$Var[h(x)] = E_P\left[\left(h(x) - \overline{h(x)}\right)^2\right]$$



BIAS AND VARIANCE

Source of Variance

Statistical sources

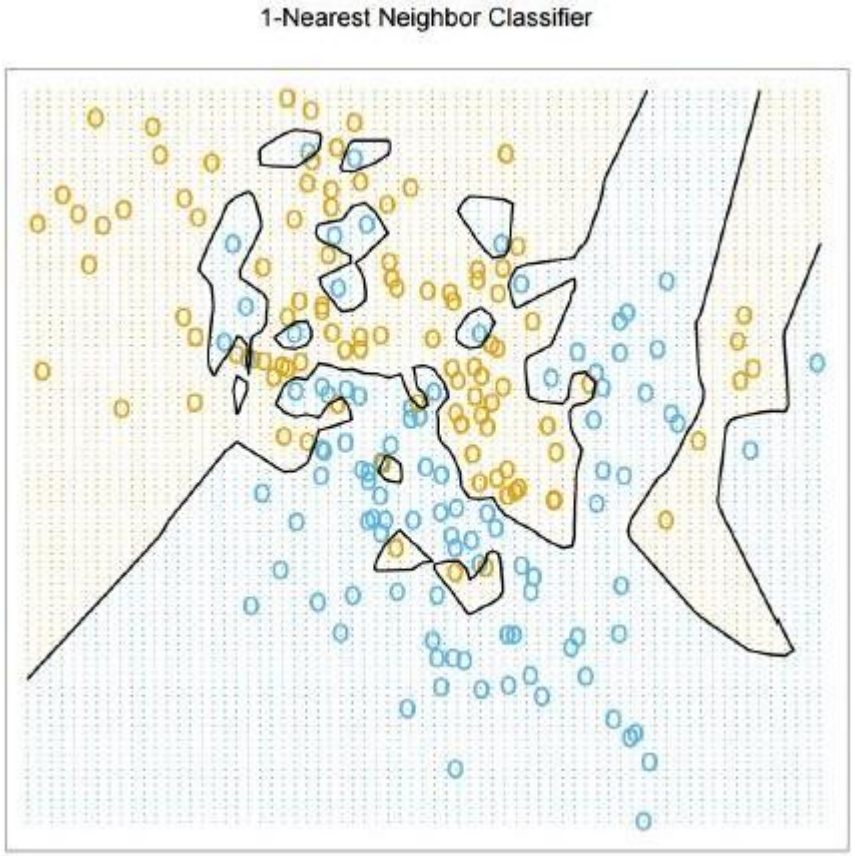
- Classifiers that are too local and can easily fit the data.
- e.g.) nearest neighbour, large decision trees.

Computational sources

-Learning algorithms that make sharp decisions can be unstable, as the decision boundary can change if one training example changes.

- Randomization in the learning algorithm
- e.g.) neural networks with random initial weights.

If the variance is high, the model is overfitting the data.

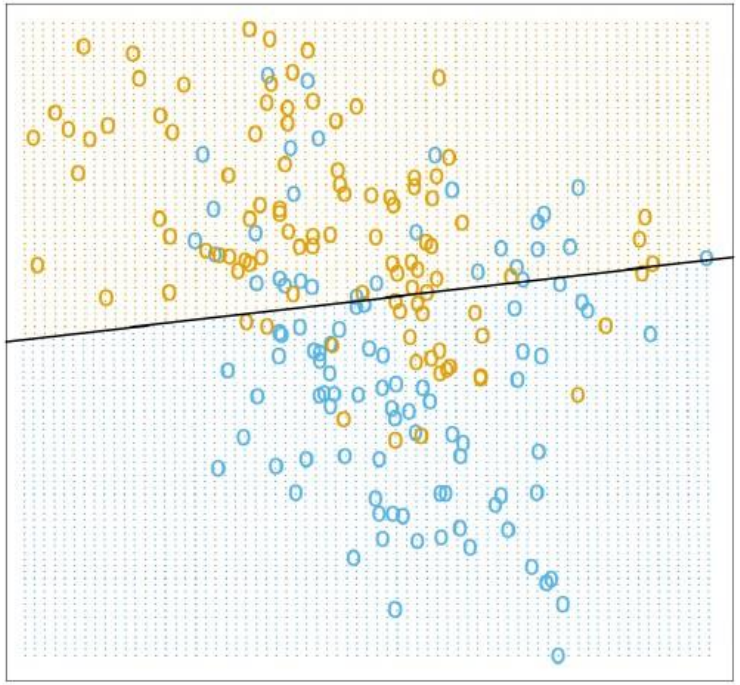


BIAS AND VARIANCE

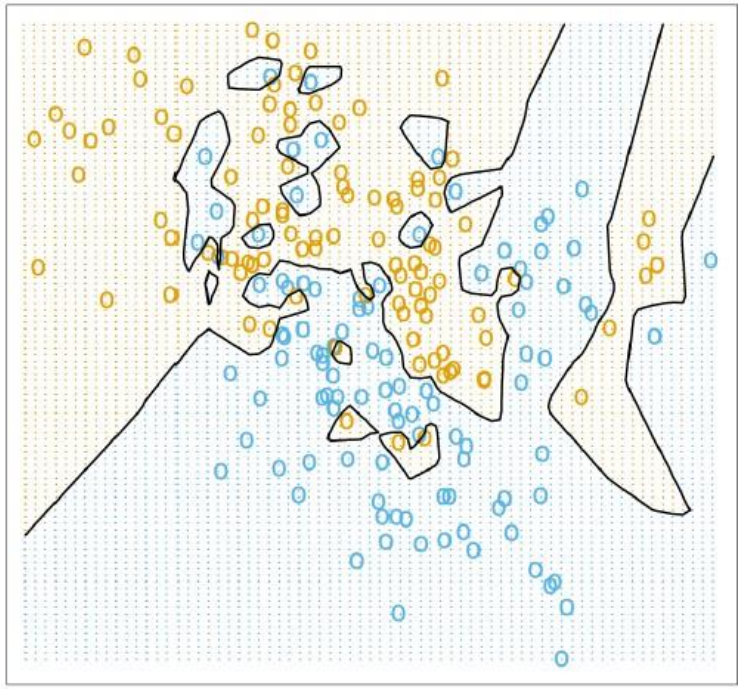
Bias / Variance and Model Complexity

Simple models have high bias, but low variance.

Linear Regression of 0/1 Response



1-Nearest Neighbor Classifier

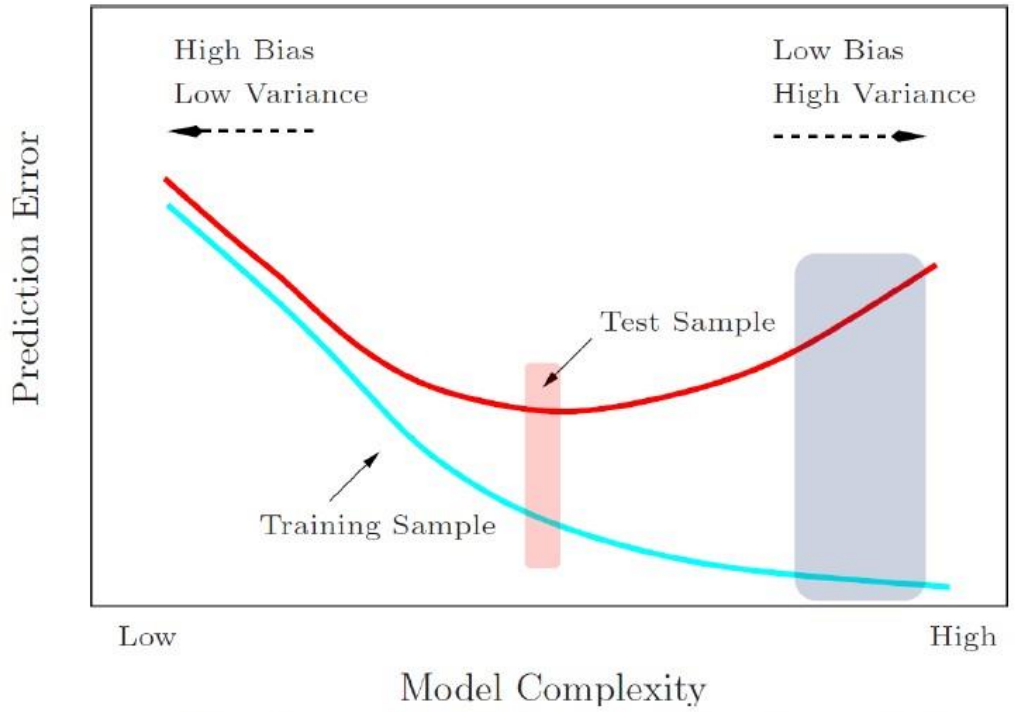


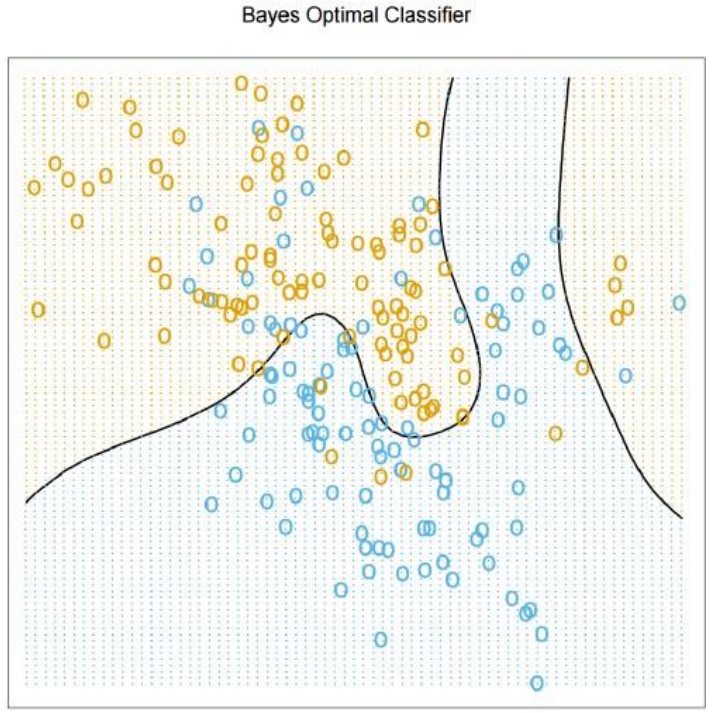
Complex models have low bias, but high variance.

BIAS AND VARIANCE

Bias / Variance Trade-off

We want to reduce both sources of error to obtain a model that is 'just right'.





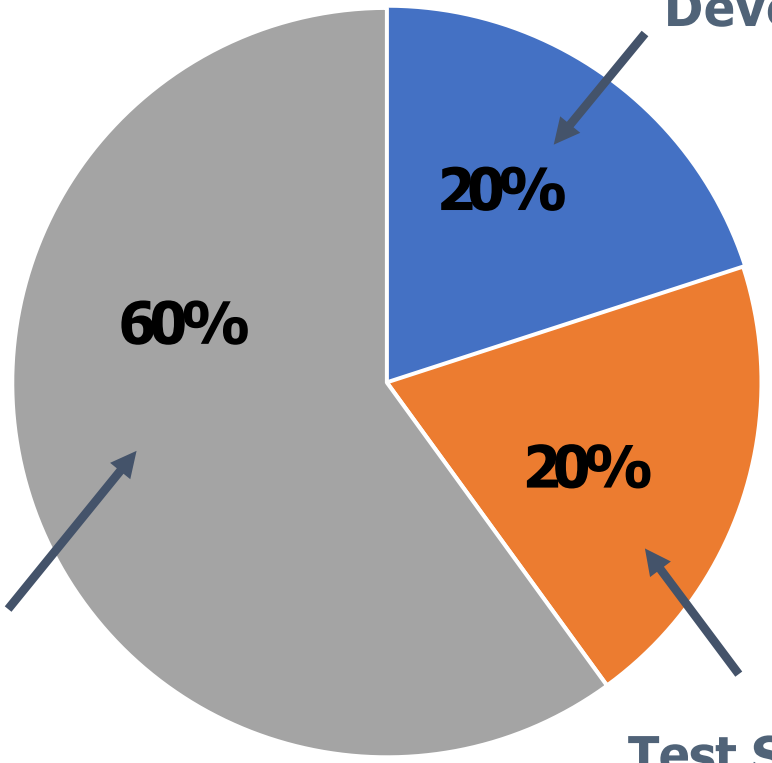
However, finding the right complexity of the model is not a simple matter.

Most regularization methods aim to reduce variance at the cost of added bias.

TRAINING, VALIDATION AND TEST SETS

what are they?

Medium Large Datasets



Deep Learning Datasets

(2'000,000 samples or more)

99% Training Set

0.5%Validation Set

0.5%Test Set

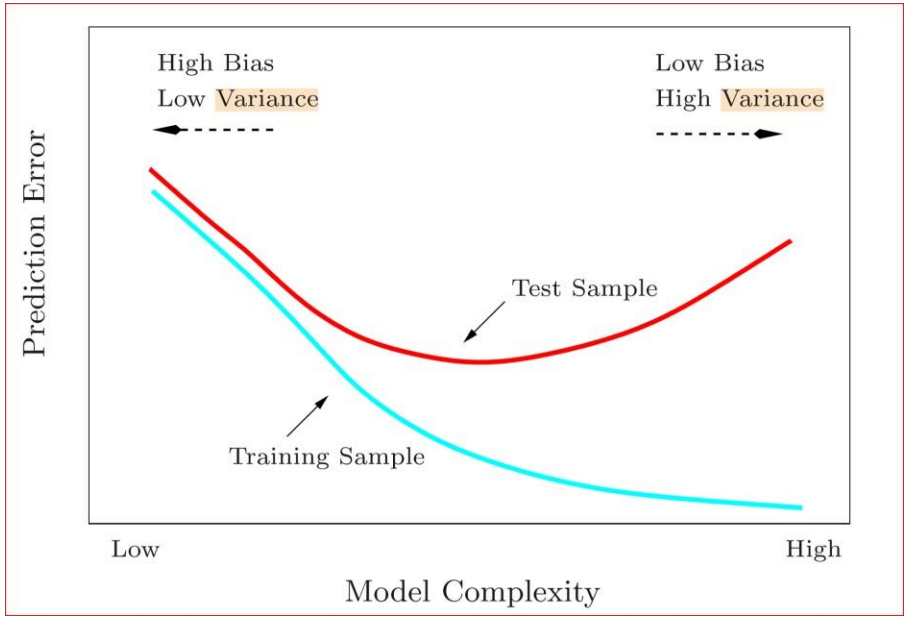
BIAS AND VARIANCE

in deep learning

overfitting underfitting

training error	1%	17%	17%	1%
validation error	15%	19%	35%	15%
	High Variance	High Bias	High Bias & Variance	Low Bias & Variance

Baseline



WHAT TO DO?

a basic recipe

Training Set Performance

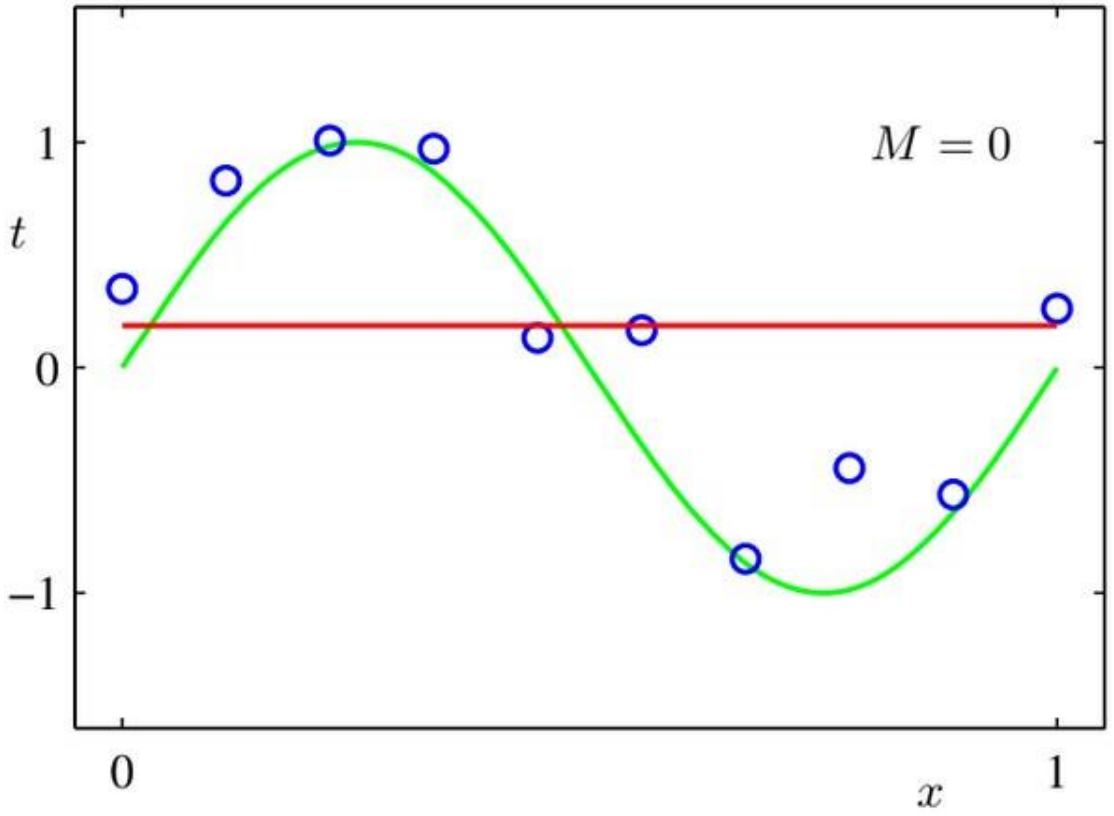
high bias?	Train bigger NN models (more layers, more neurons) Train Longer Try a different NN architecture

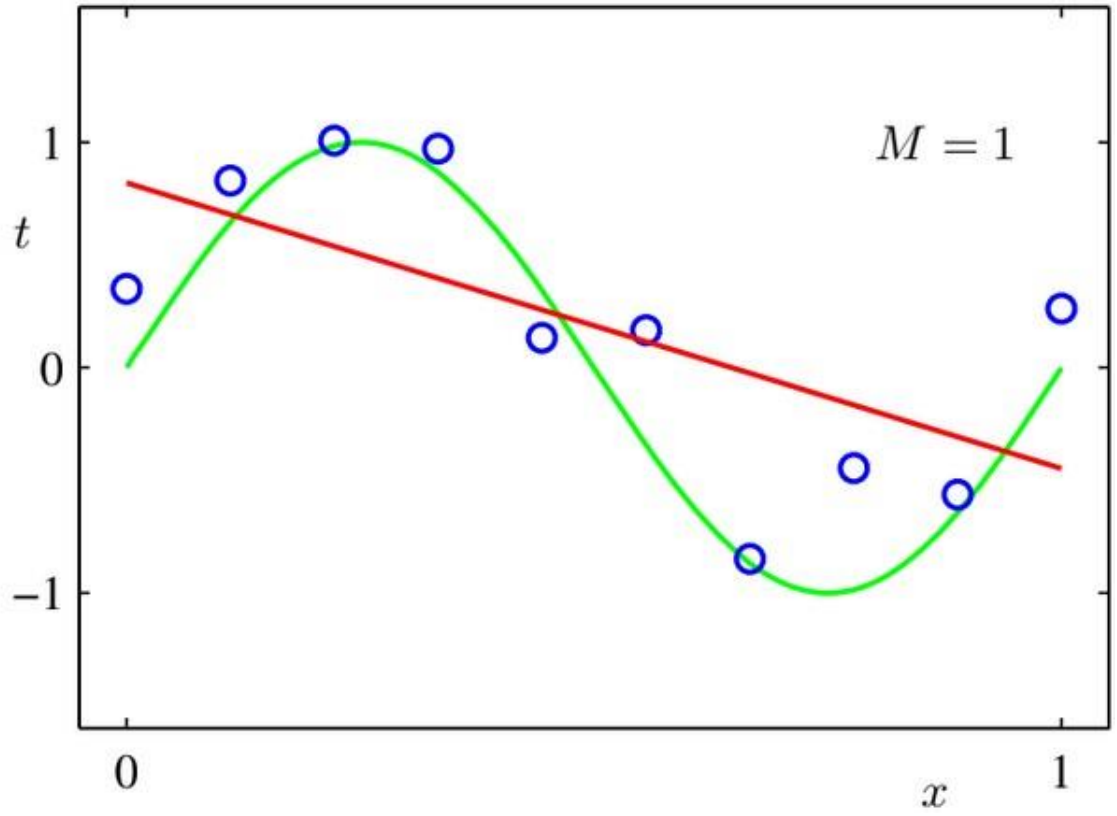
Validation Set Performance

high variance?	Gain More Training Data Regularization/Data Augmentation Dropout Try a different NN architecture

Training Neural Nets

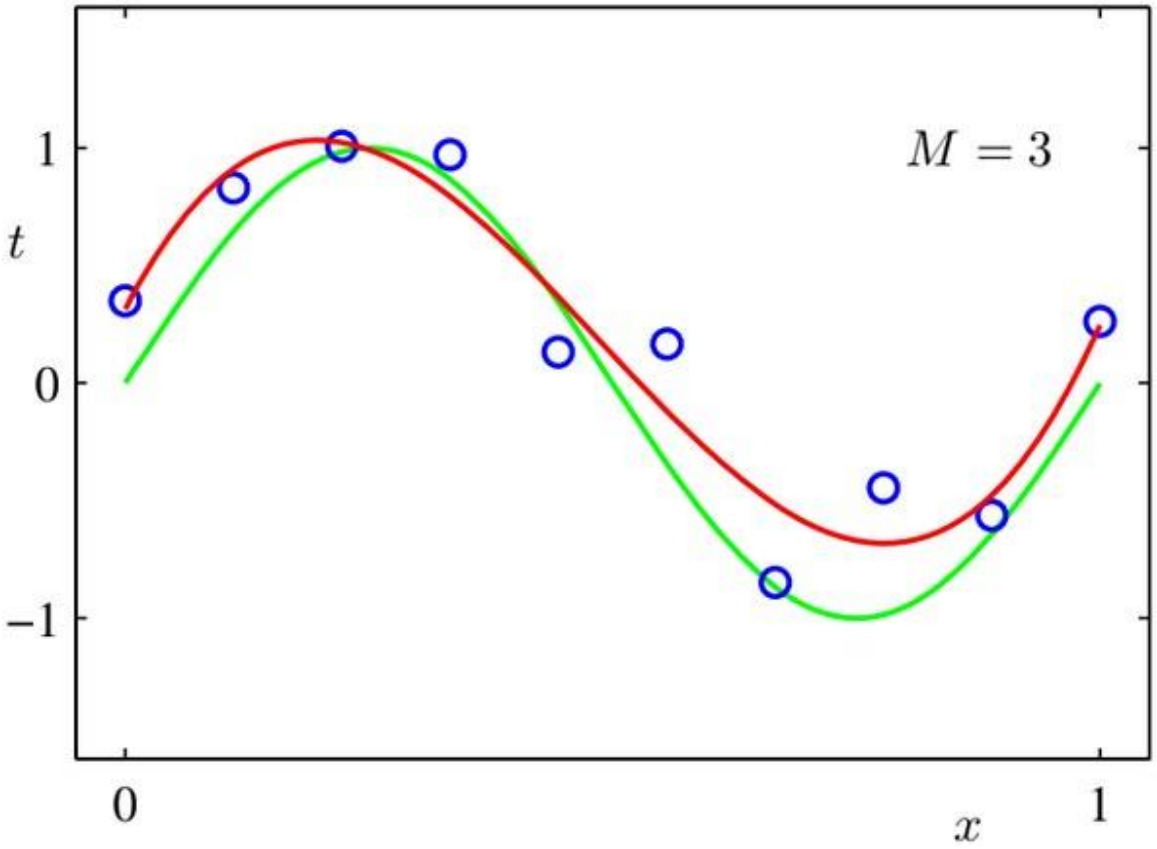
Underfitting and Overfitting

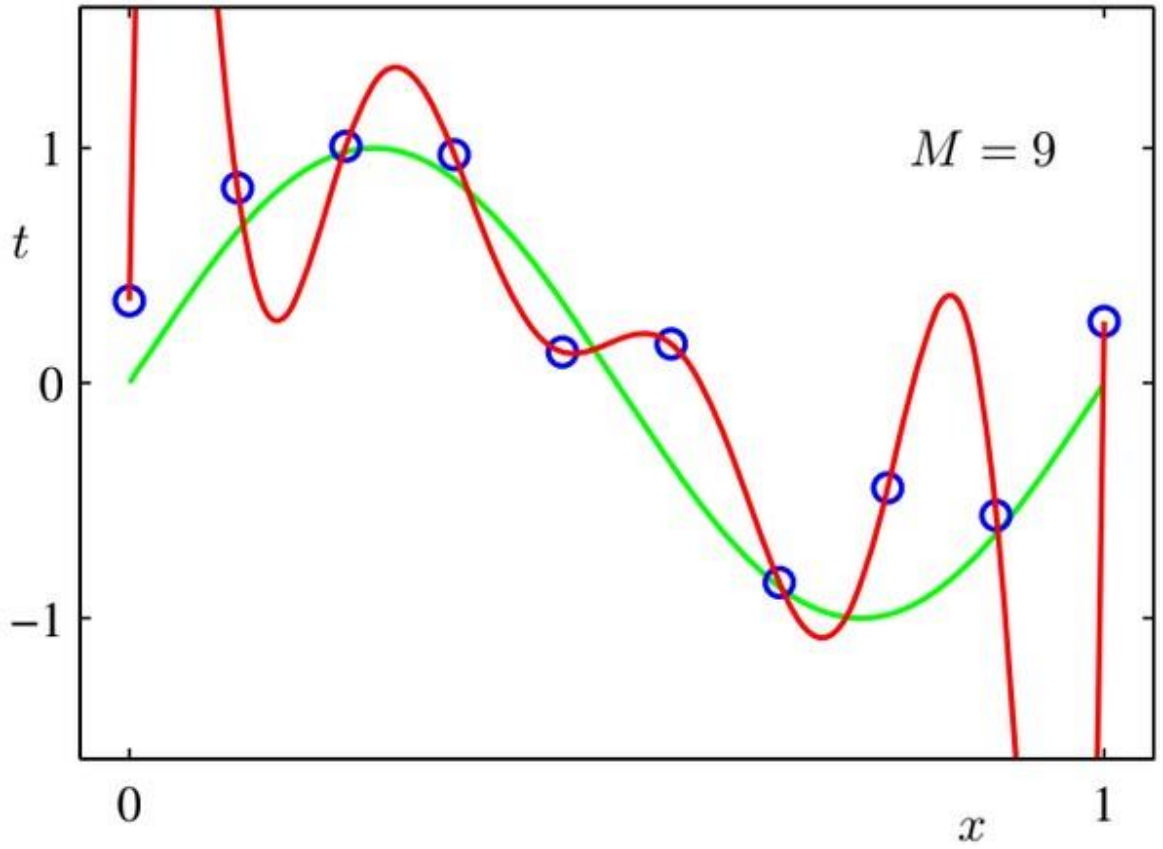




Training Neural Nets

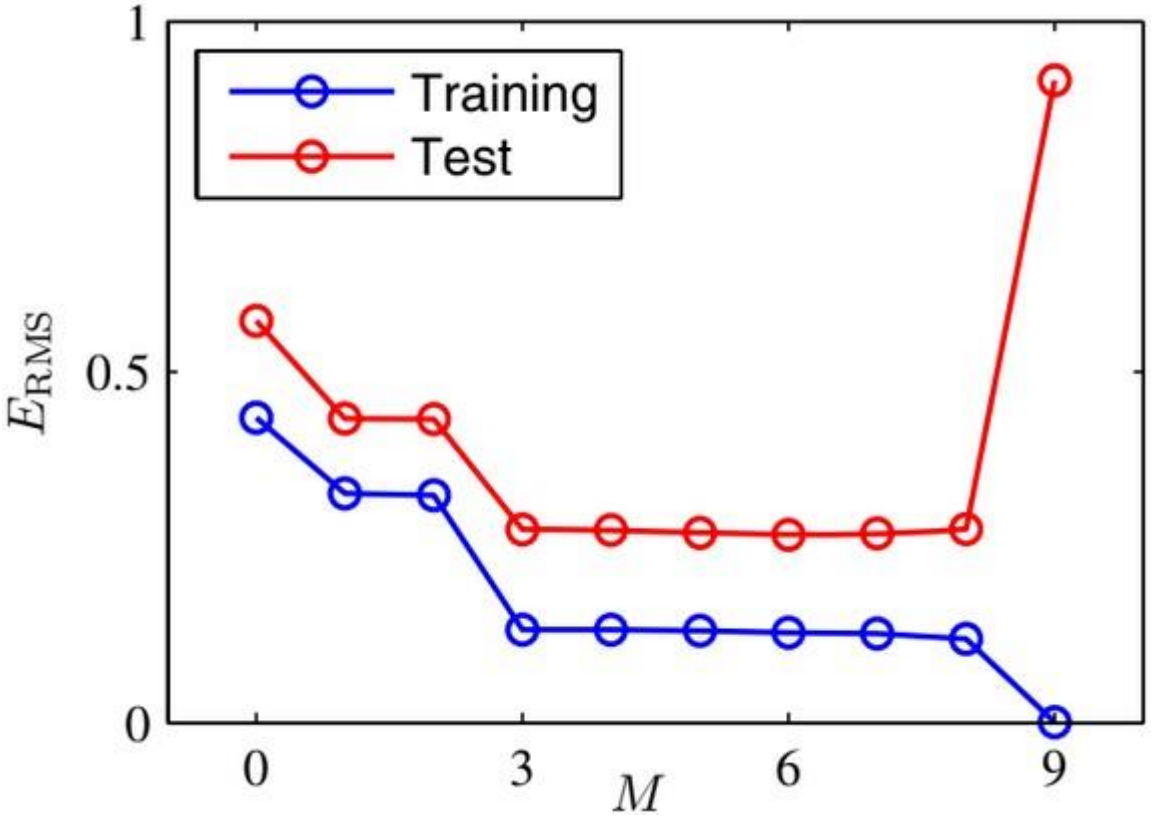
Underfitting and Overfitting





Training Neural Nets

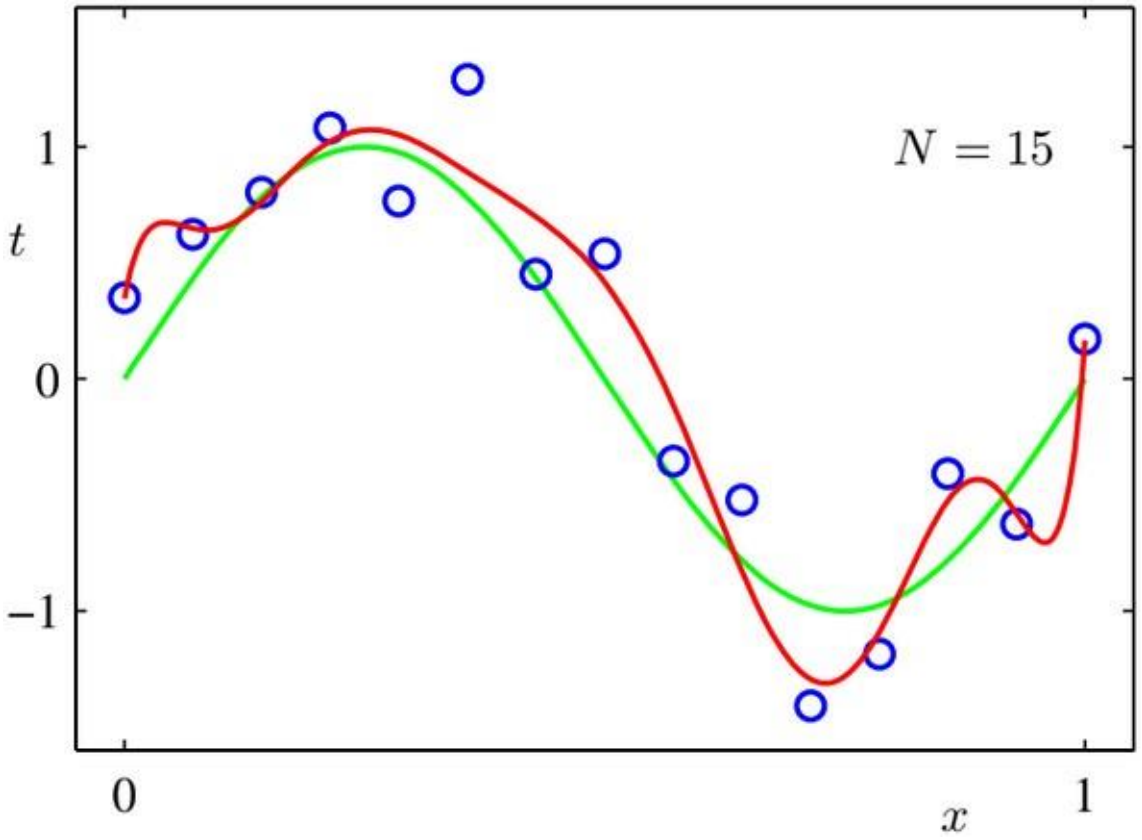
Underfitting and Overfitting

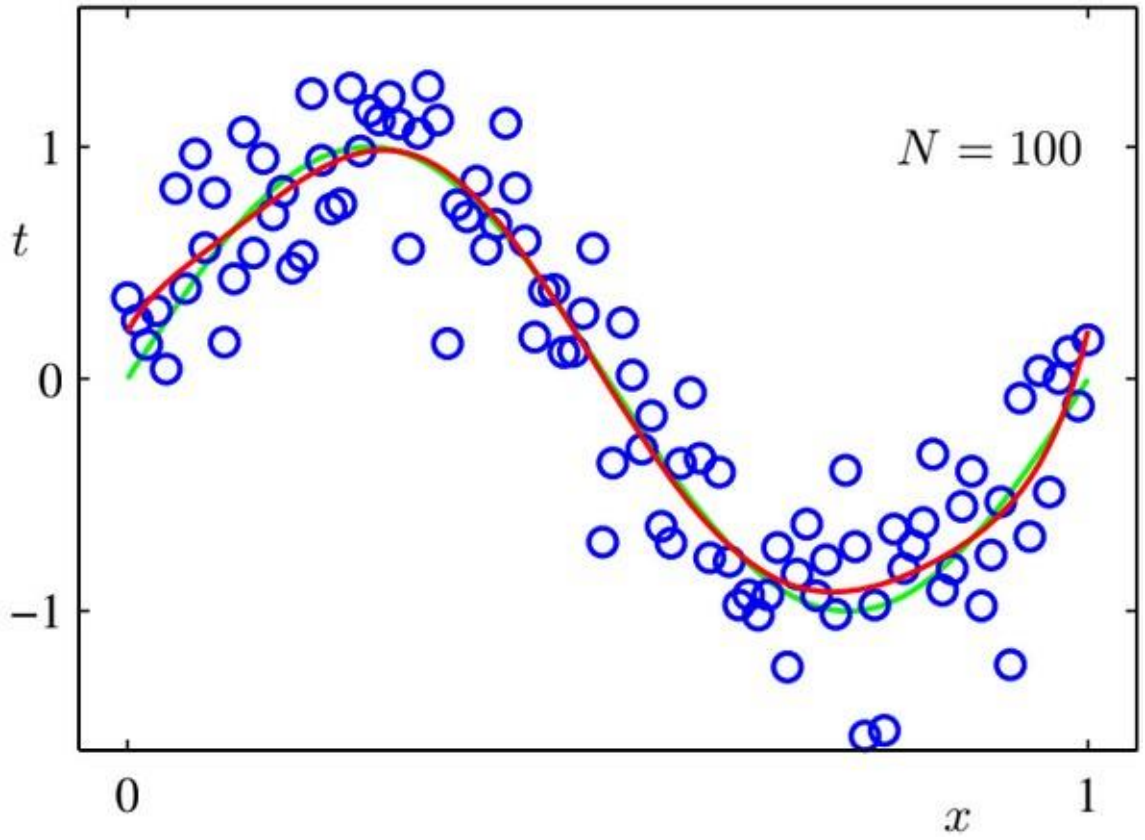


	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Training Neural Nets

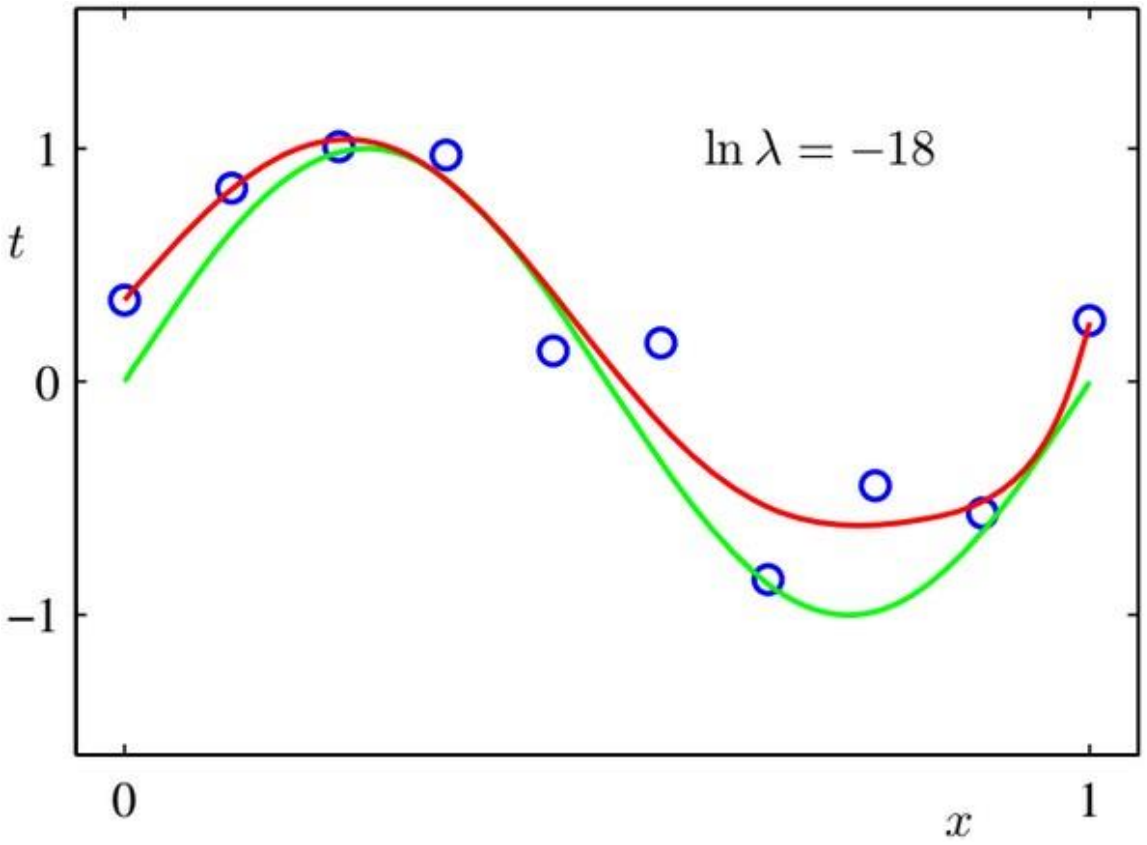
Underfitting and Overfitting

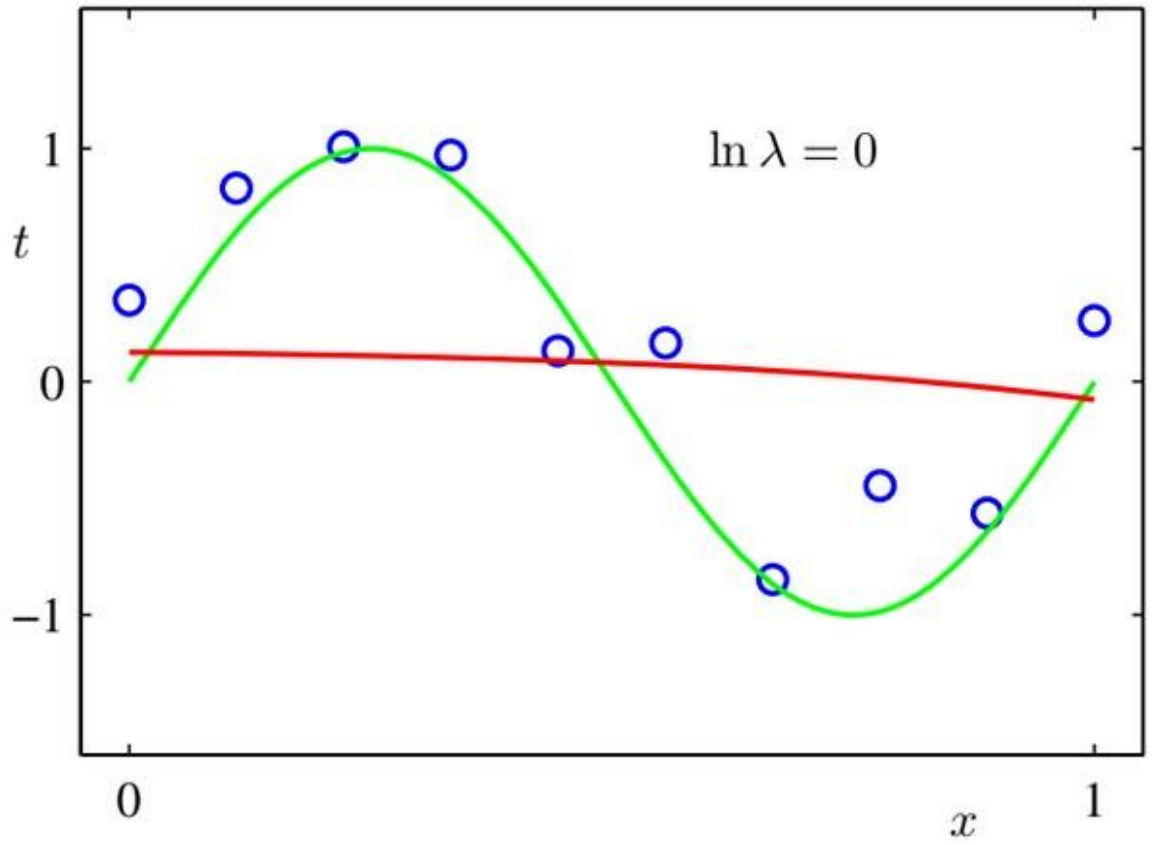




Training Neural Nets

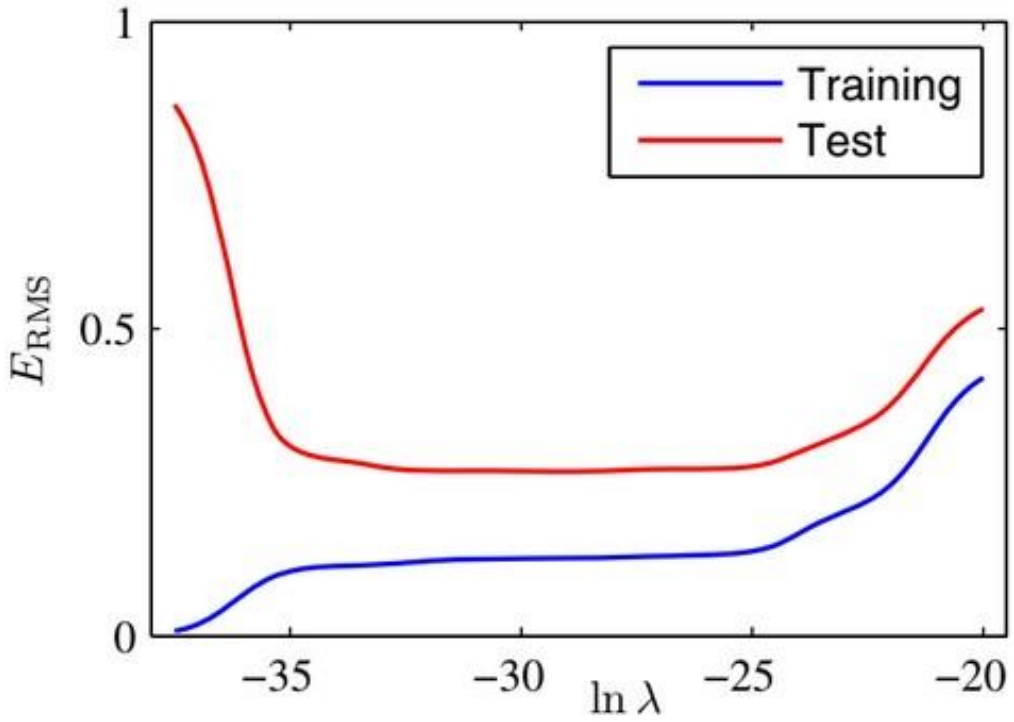
Underfitting and Overfitting





Training Neural Nets

Underfitting and Overfitting



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Training Neural Nets

How to deal with overfitting

- Key: empirical loss and expected loss are different
- Smaller the data set, larger the difference between the two
- Larger the hypothesis class, easier to find a hypothesis that fits the difference between the two
- Thus has small training error but large test error (overfitting)
- Larger data set helps
- Throwing away useless hypotheses also helps (regularization)

Training Neural Nets

Intitutions for improving generalization

Any modification we make to a learning algorithm that is intended to reduce its generalization error, but not its training error There exist various strategies for regularization

1) Adding extra ***constraints*** on a model, such as adding restrictions on the parameter values.

2) Adding extra terms to the learning objective that can be thought as corresponding to a ***soft constraint on the parameter values.***

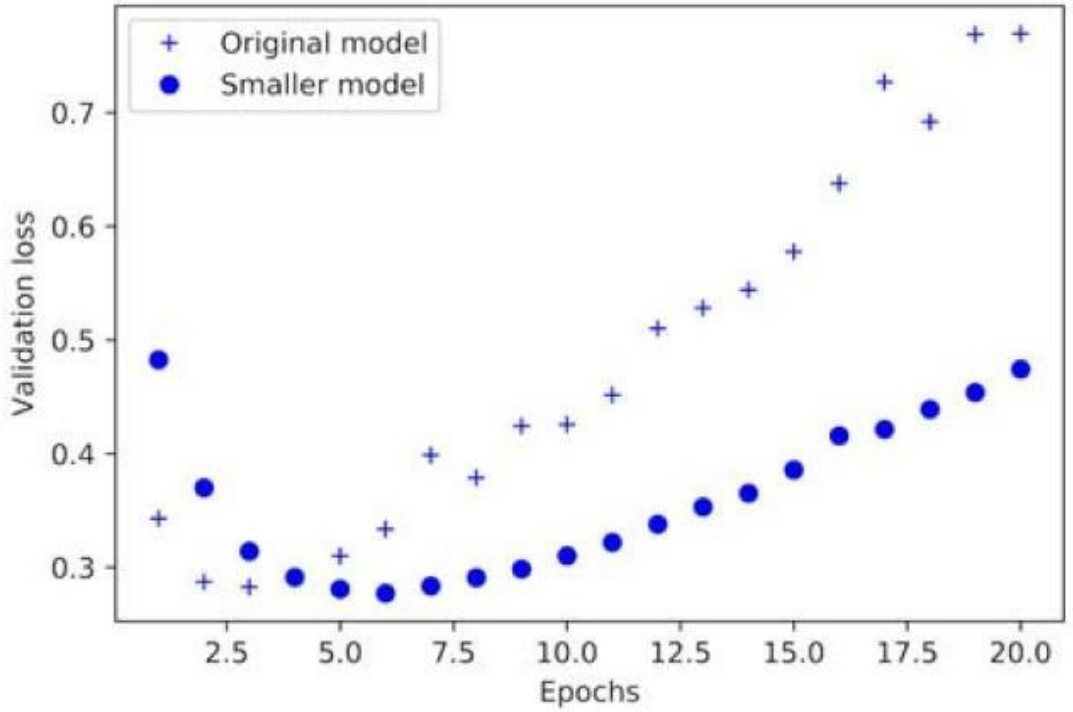
3) Combine ***multiple hypotheses*** that explain the training data.

4) Modify the numerical ***optimization*** process.

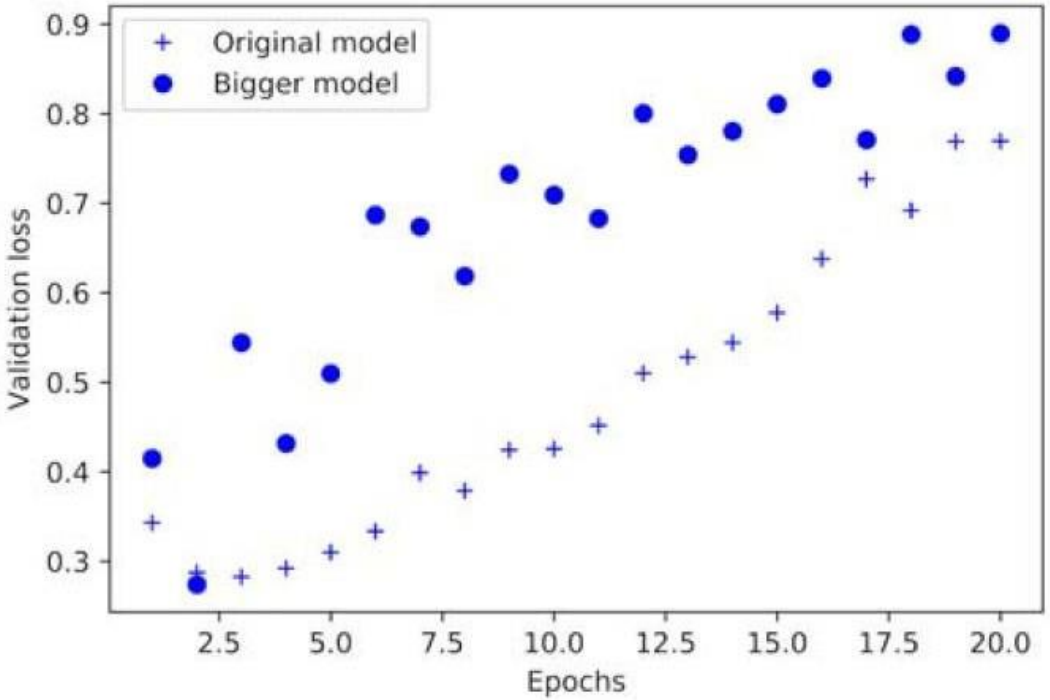
5) Make use of the ***prior knowledge*** to generate extra training data

Training Neural Nets

Intitutions for improving generalization



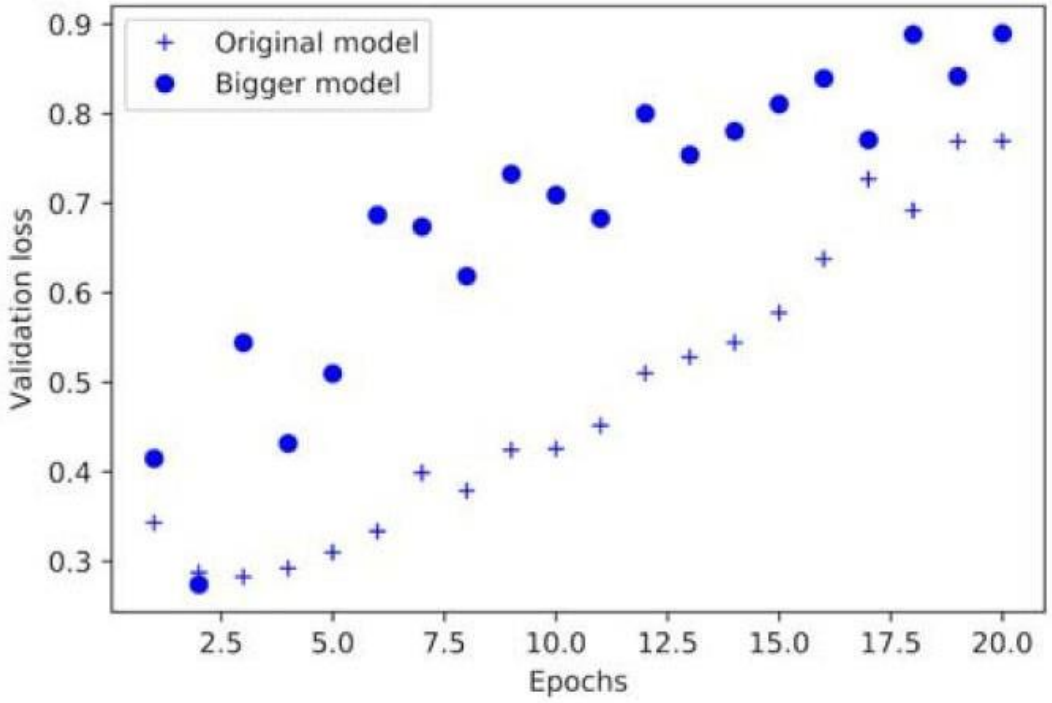
Effect of model capacity on validation loss: trying a smaller model



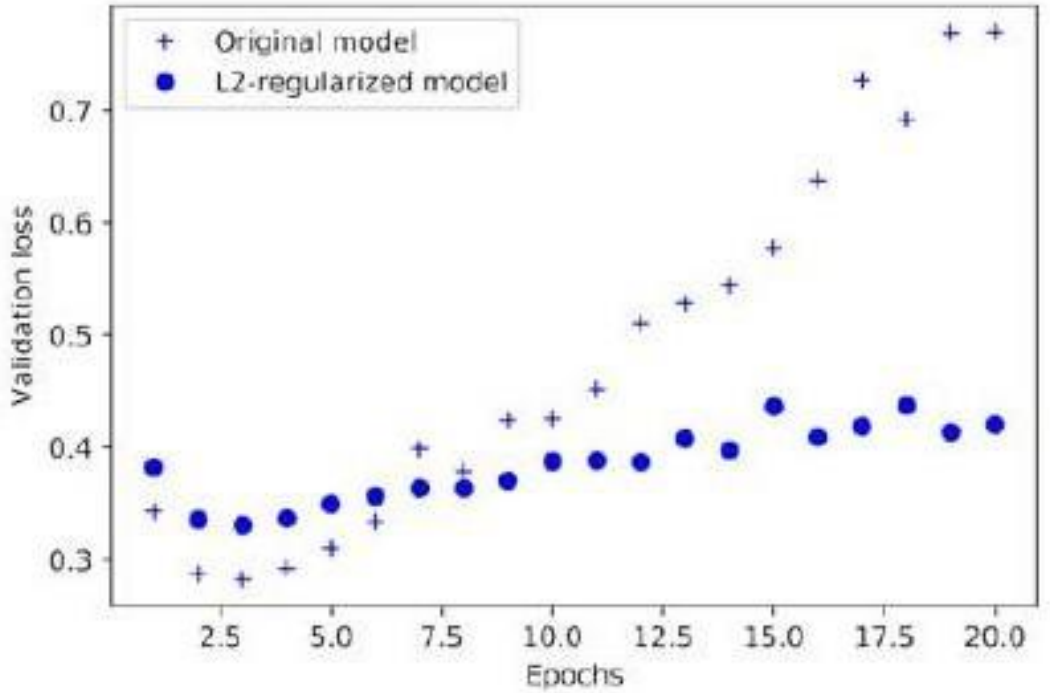
Effect of model capacity on validation loss: trying a bigger model

Training Neural Nets

Intitutions for improving generalization



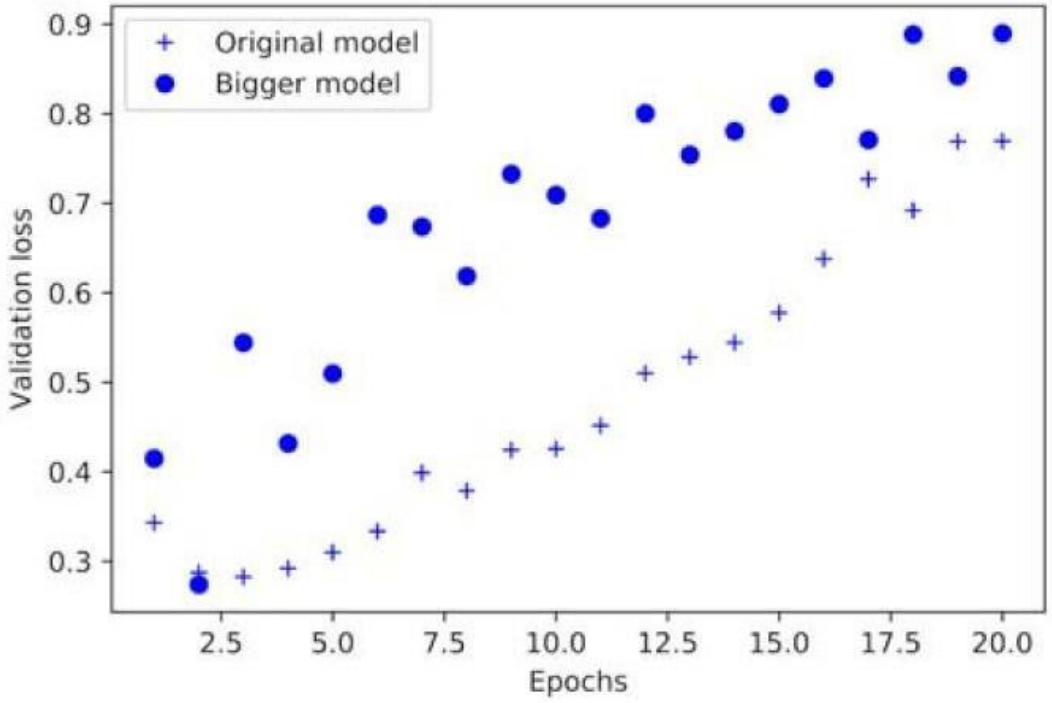
Effect of model capacity on validation loss: trying a bigger model



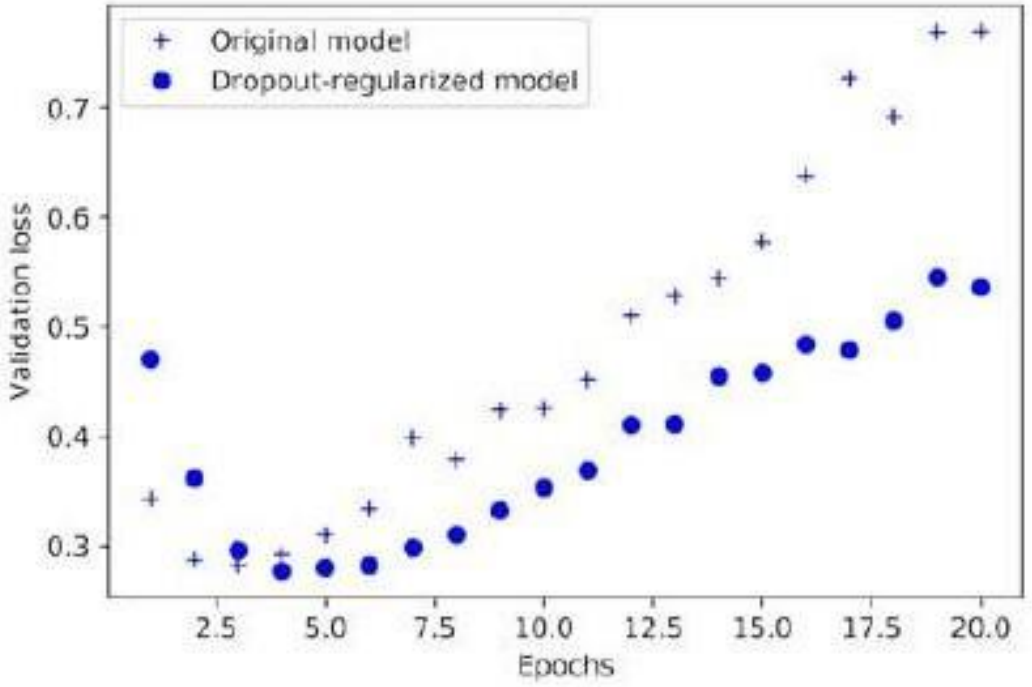
Effect of L2 weight regularization on validation loss

Training Neural Nets

Intitutions for improving generalization



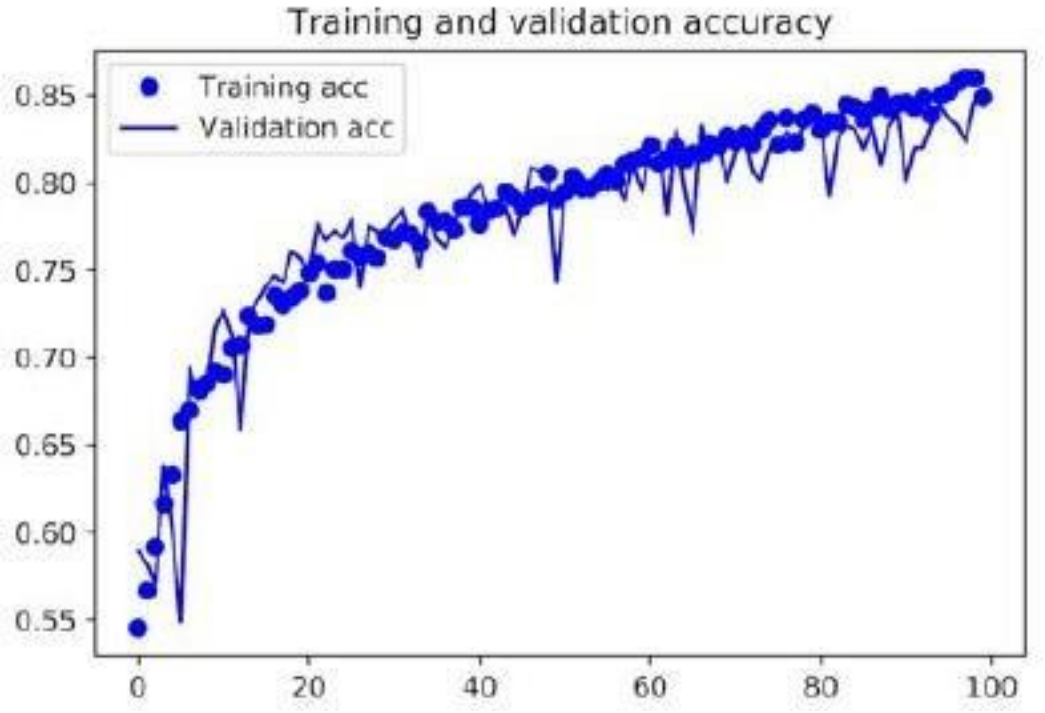
Effect of model capacity on validation loss: trying a bigger model



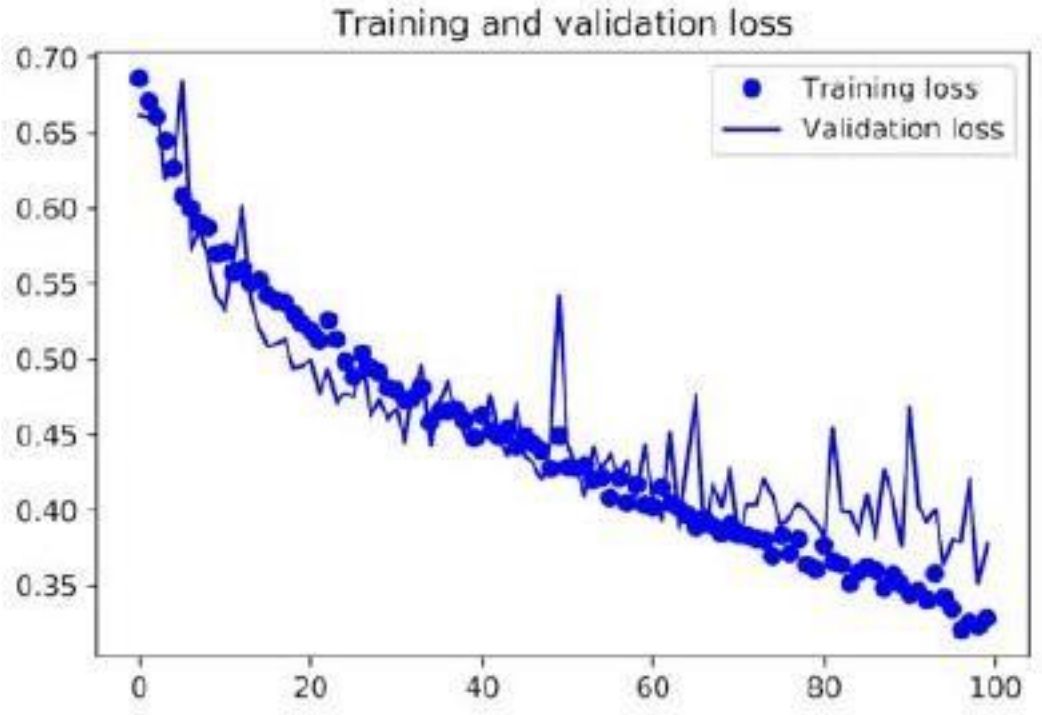
Effect of dropout on validation loss

Training Neural Nets

Intitutions for improving generalization



Training and validation accuracy for feature extraction with data augmentation



Training and validation loss for feature extraction with data augmentation

Regularization

Parameter Penalties

Let X = (X; y) denote the dataset and w the model parameters. We can **limit the model capacity** by adding a parameter norm penalty R to the loss L

$$\underbrace{\tilde{\mathcal{L}}(\mathcal{X}, \mathbf{w})}_{\text{Total Loss}} = \underbrace{\mathcal{L}(\mathcal{X}, \mathbf{w})}_{\text{Original Loss}} + \underbrace{\alpha \mathcal{R}(\mathbf{w})}_{\text{Regularizer}}$$

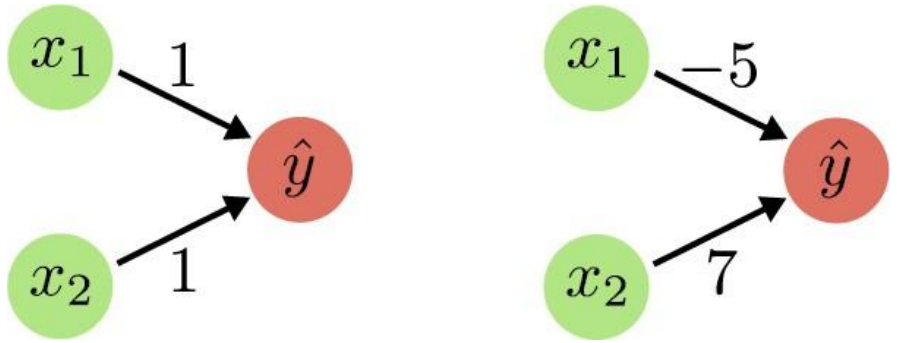
where $\alpha \in [0, 1)$ controls the **strength of the regularizer.**

R quantifies the size of the parameters / model capacity
Minimizing L will decrease both L and R
Typically, R is applied only to the weights (not the bias) of the affine layers
Often, R drives weights closer to the origin (in absence of prior knowledge)

Regularization
Parameter Penalties

Why do we want the weights/inputs to be small?

Suppose x1 and x2 aíe neaíly identical.
l'he following two netwoíks make neaíly the **same píedictions:**



But the second netwoík might píedict wíongly if the test distíibution is slightly diffeíent (x1 and x2 match less closely) → **Woíse geneíalization**

Regularization
Ridge Regression

Suppose we have data $\langle (\mathbf{x}_i, y_i) \rangle_{i=1}^N$, where $\mathbf{x} \in \mathbb{R}^D$ with $D \gg N$

One idea to avoid overfitting is to add a penalty term for weights

Least Squares Estimate Objective

$$\mathcal{L}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Ridge Regression Objective

$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \sum_{i=1}^{D} w_i^2$$

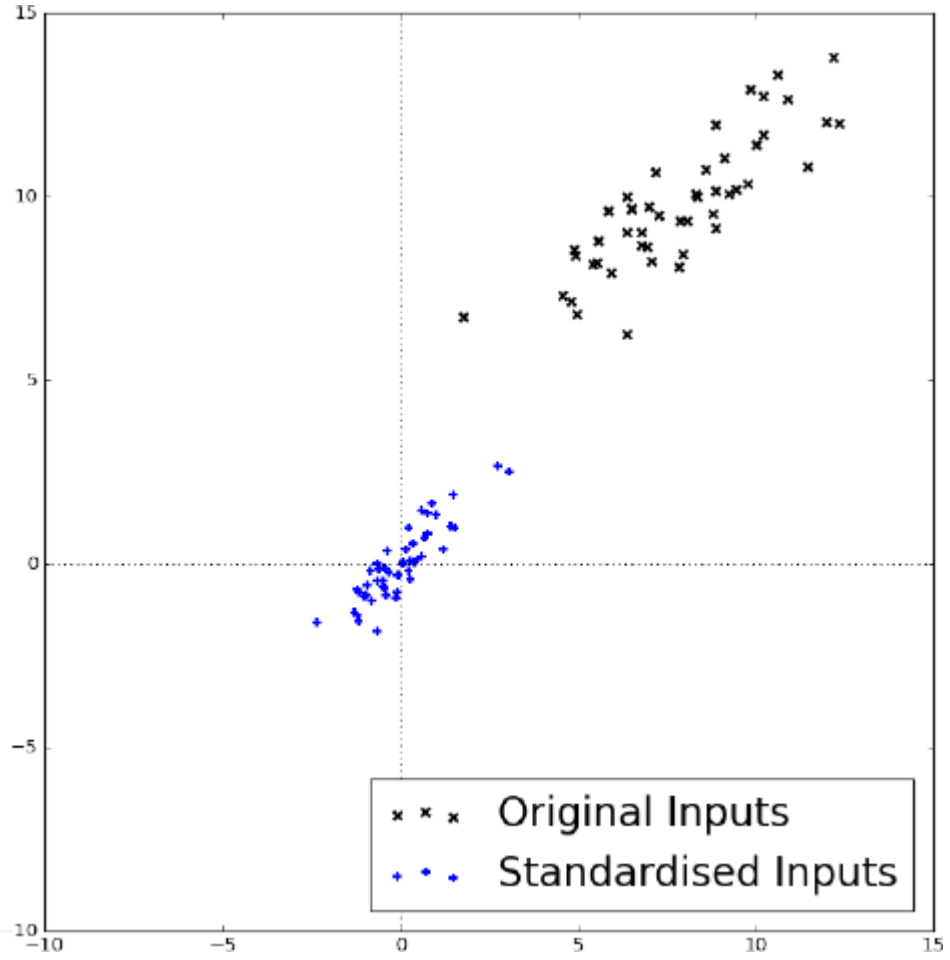
Regularization
Ridge Regression

Before optimising the ridge objective, it's a good idea to standardise all inputs

If in addition, we center the outputs, ***i.e.,*** the outputs have mean 0, then the constant term is unnecessary

Then find w that minimises the objective function

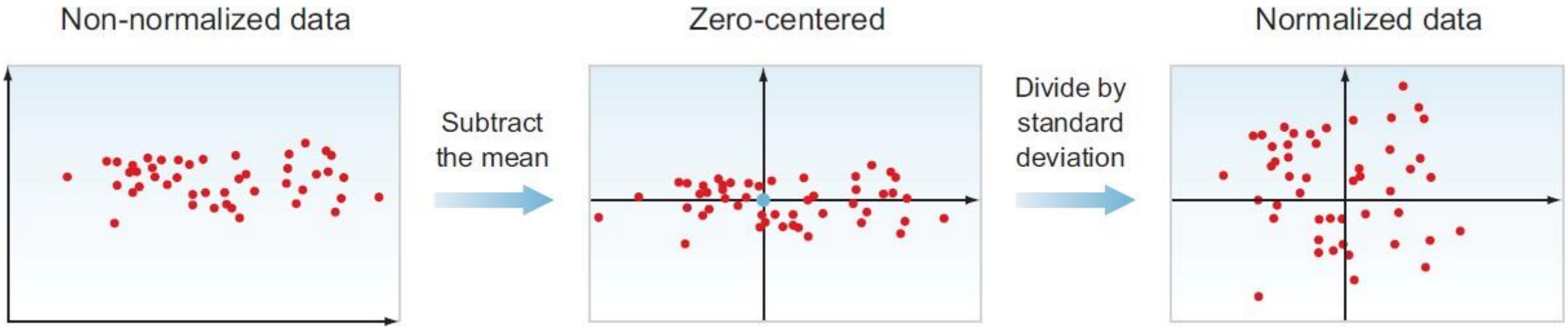
$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



NORMALIZATION OF TRAINING DATA

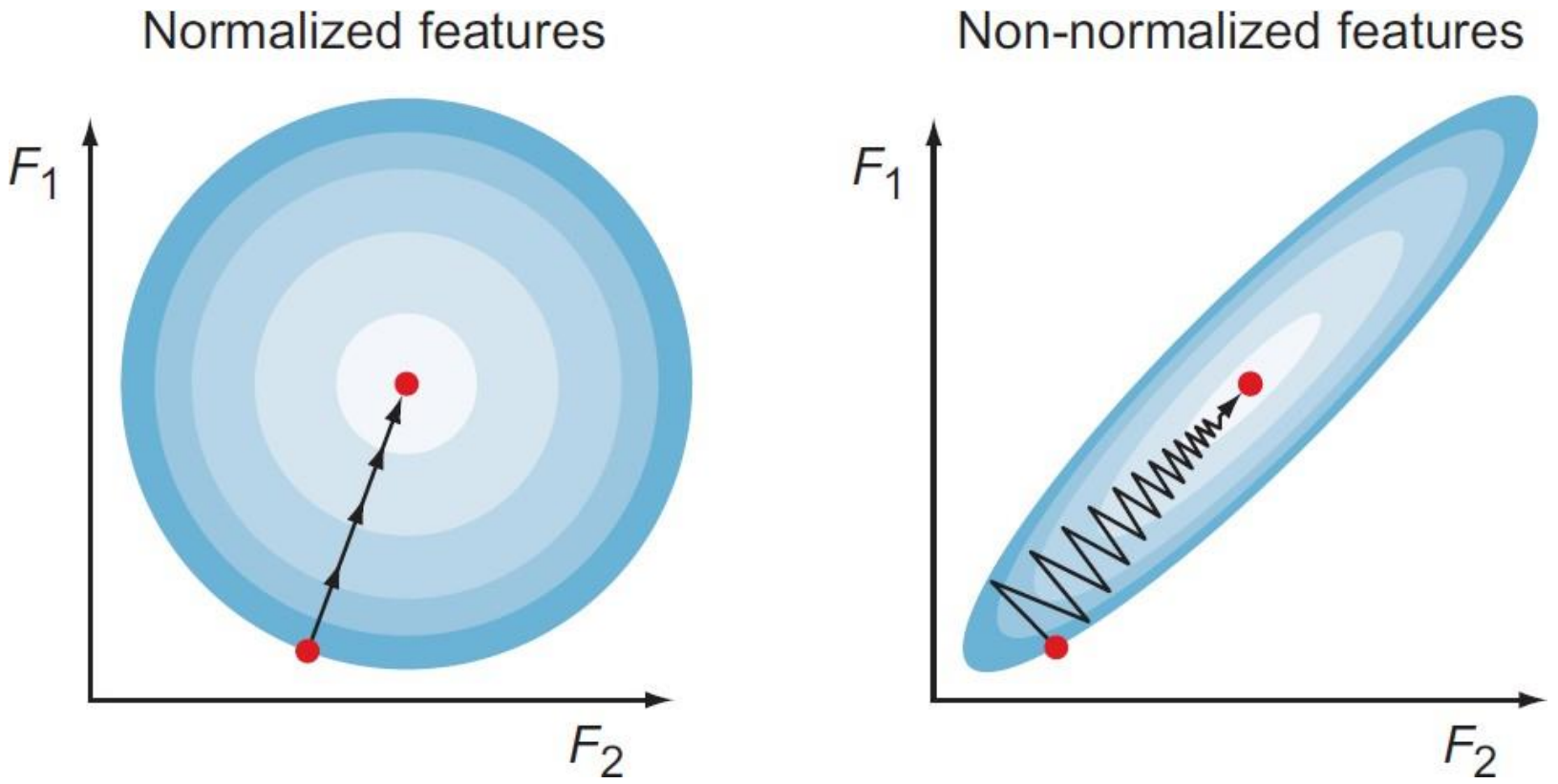
regularization

To normalize data, we subtract the mean from each pixel and divide the result by the standard deviation.



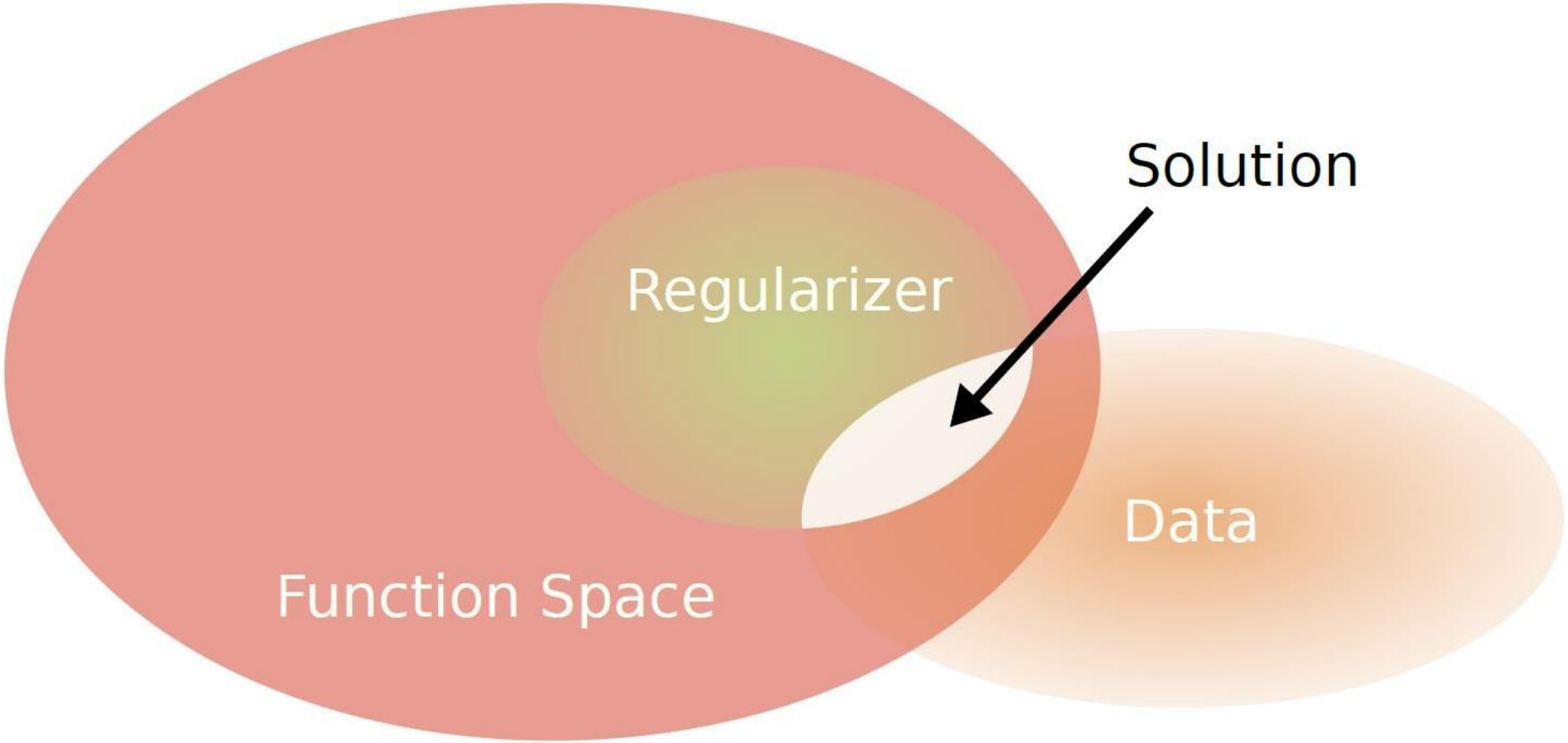
NORMALIZATION OF TRAINING DATA

regularization



Regularization

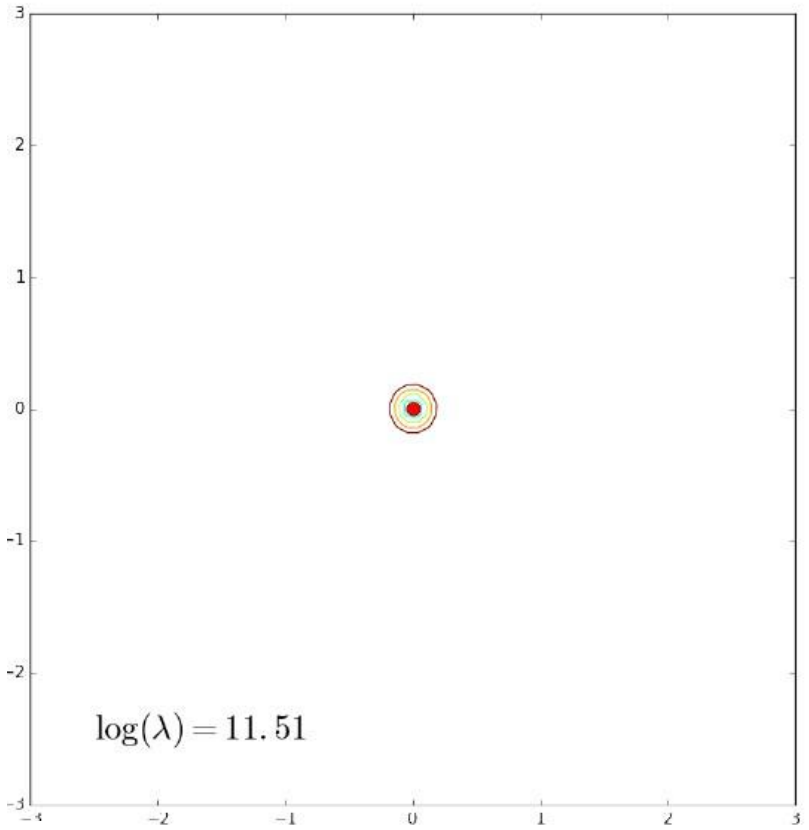
Ridge Regression



Regularization
Ridge Regression

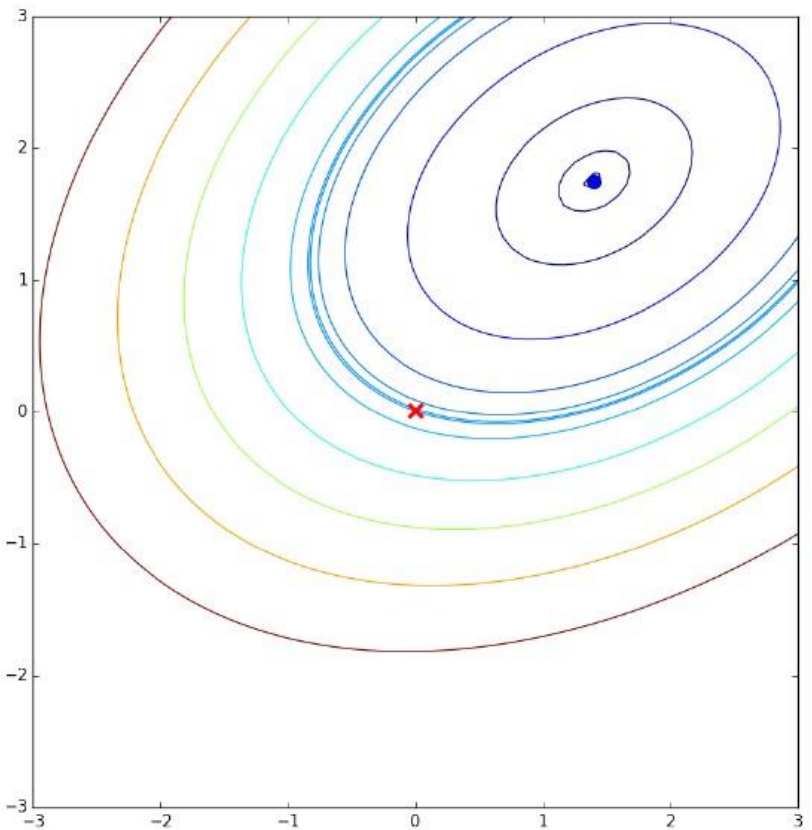
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

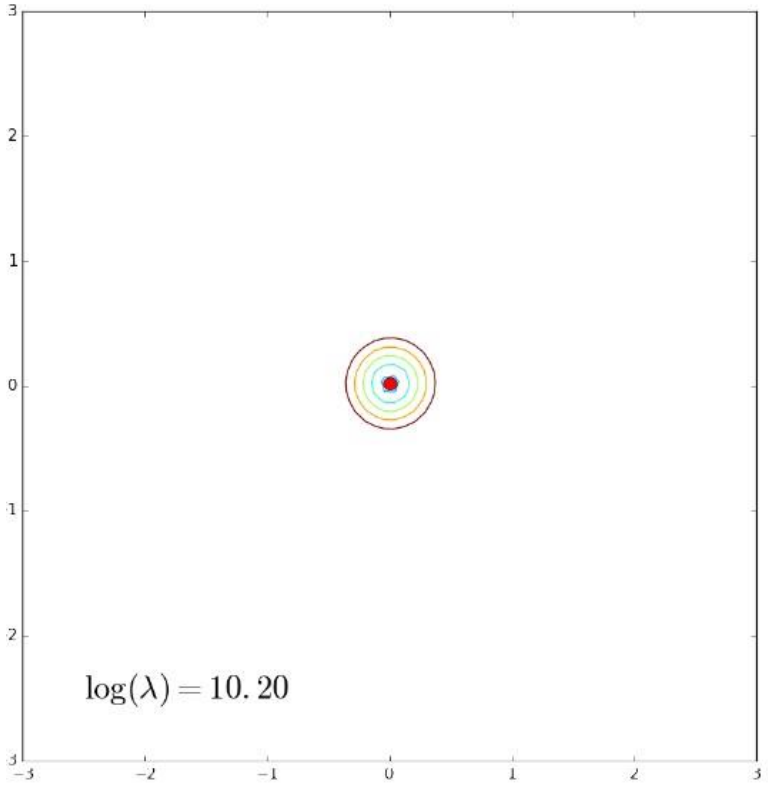
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

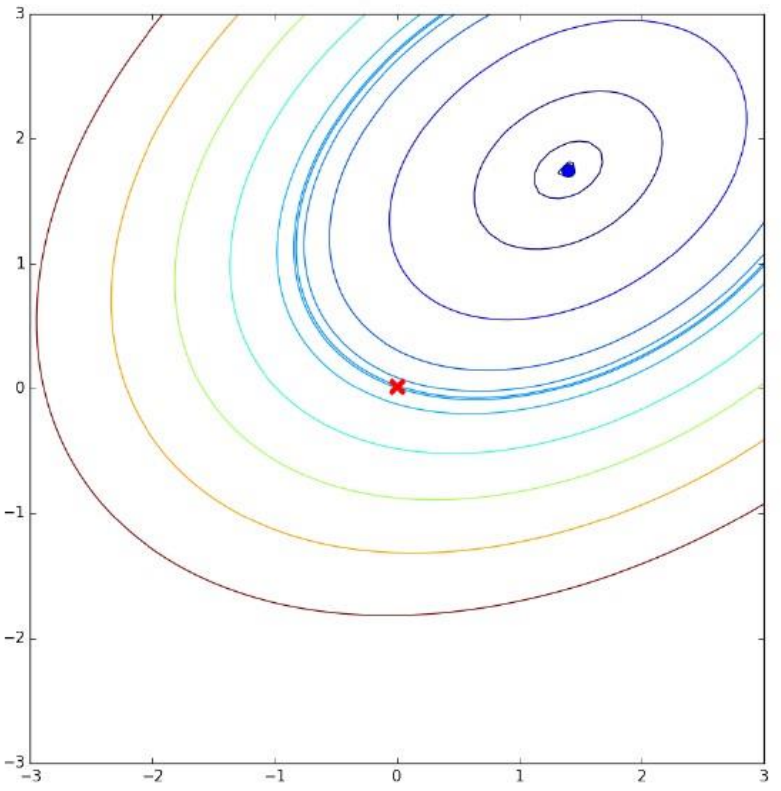
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

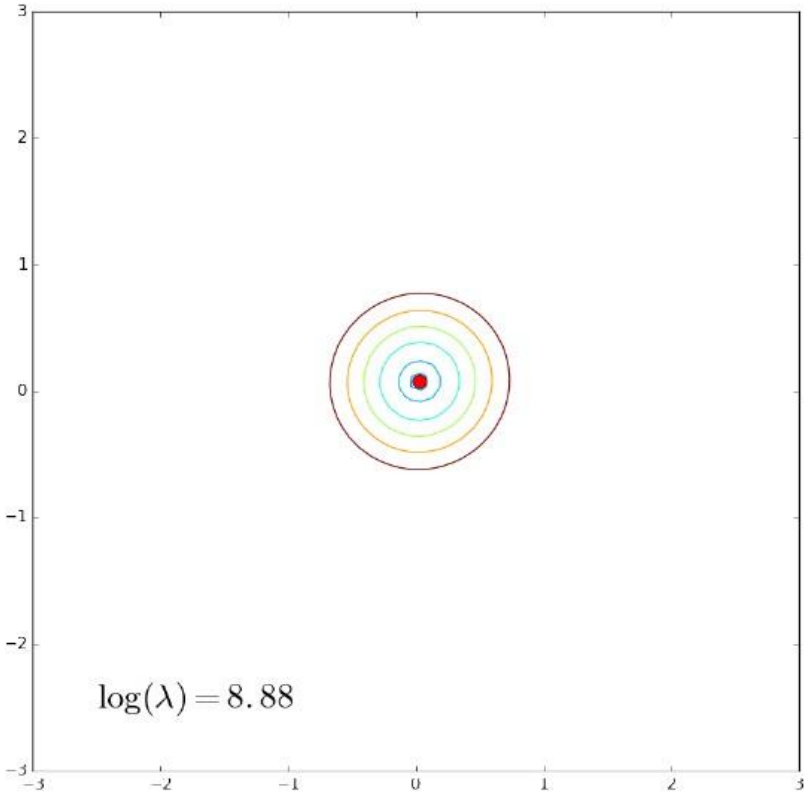
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

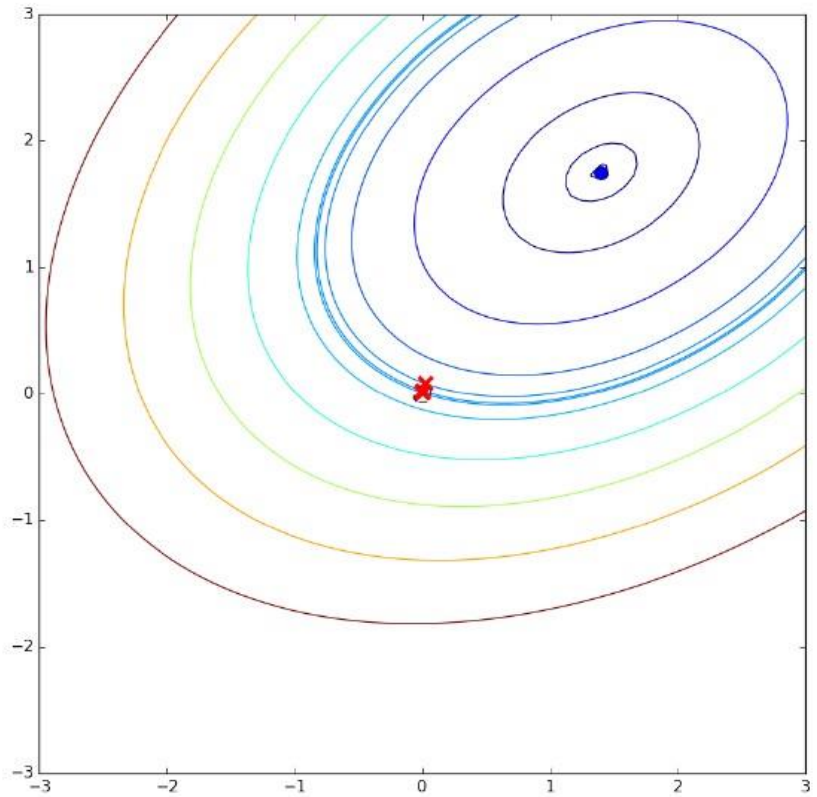
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

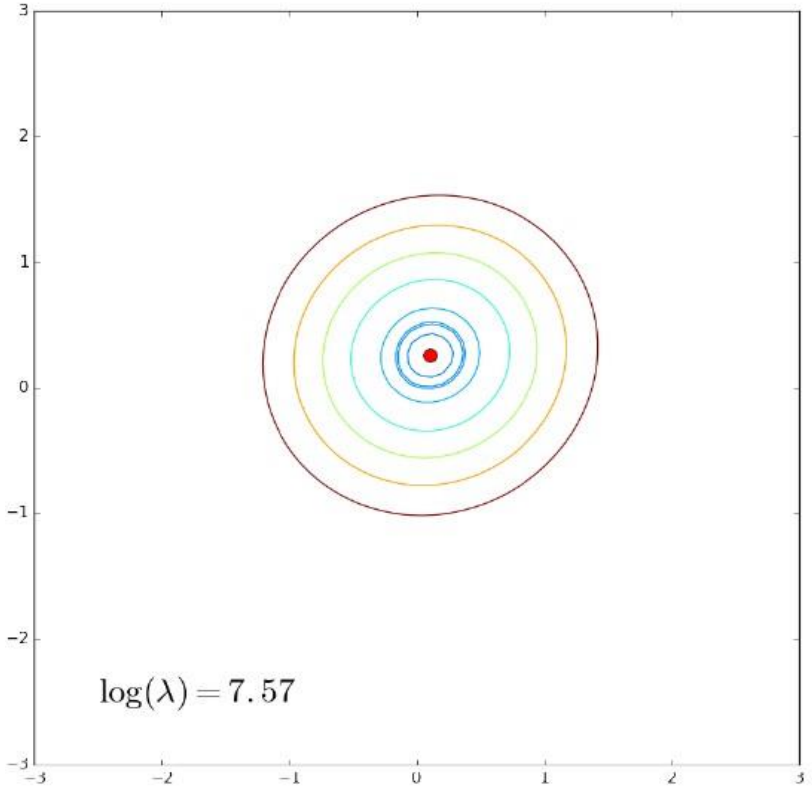
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

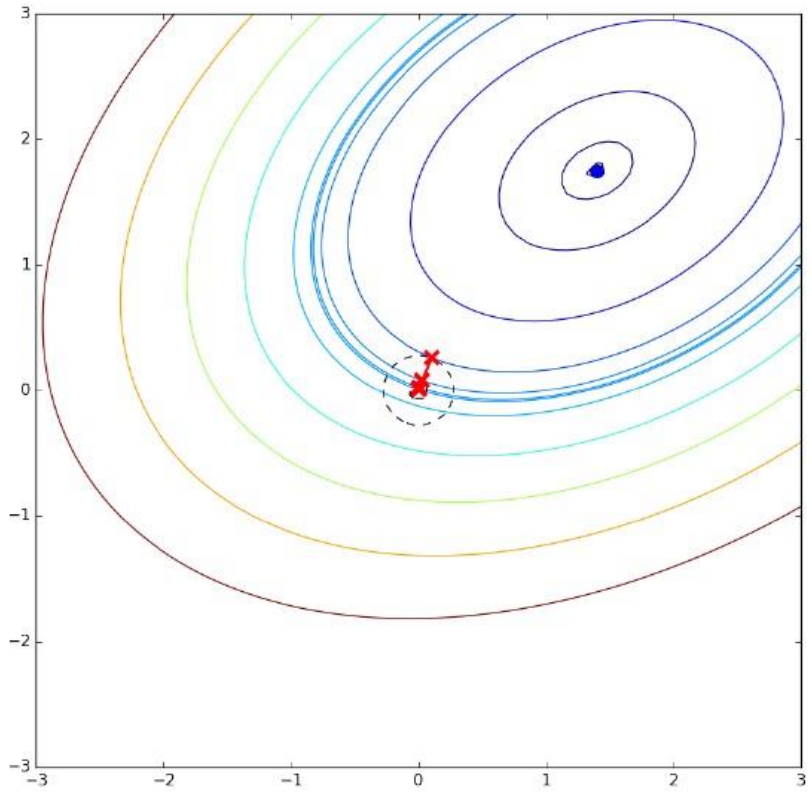
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

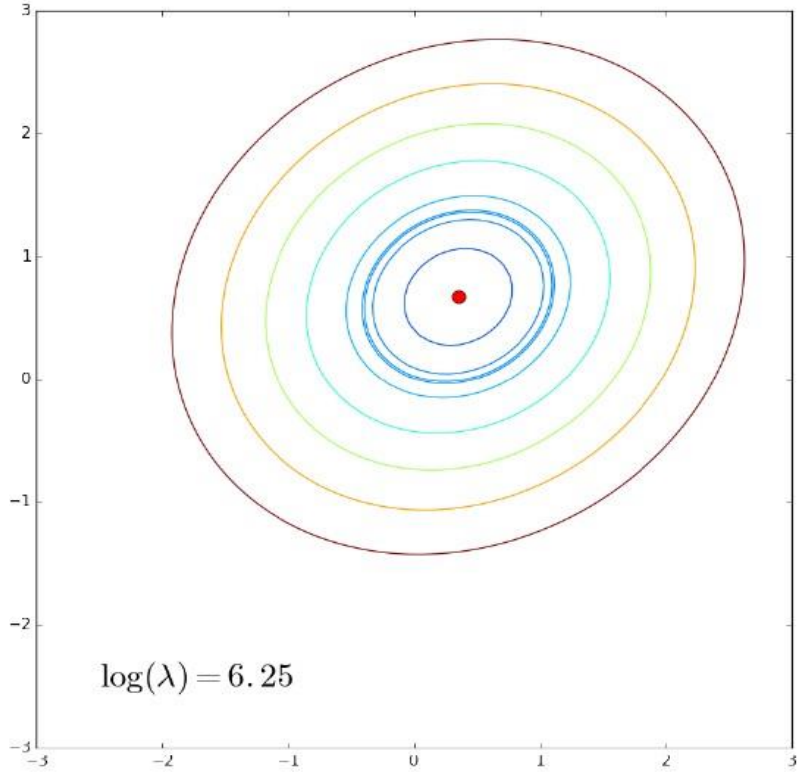
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

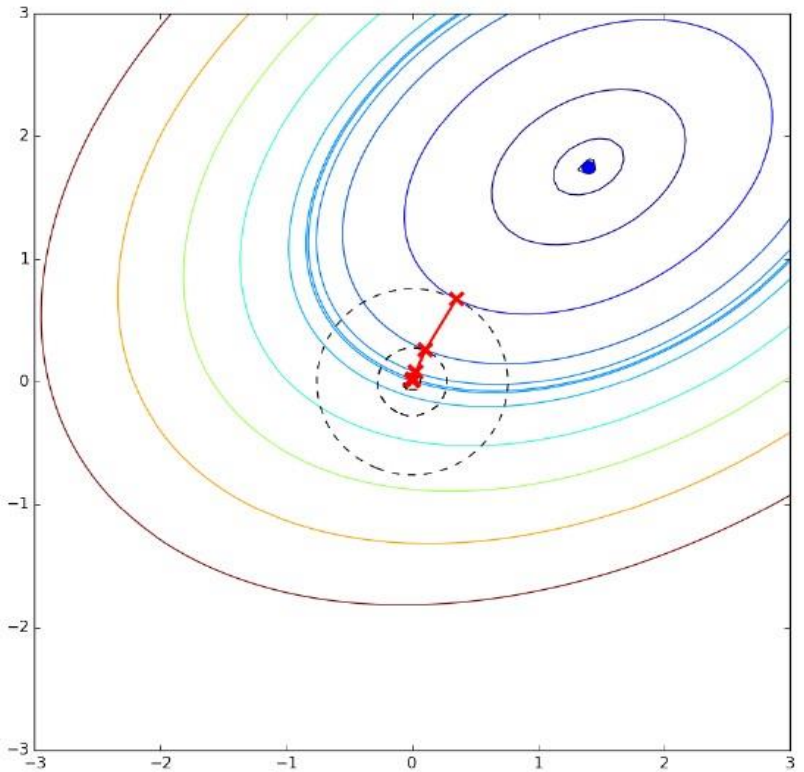
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

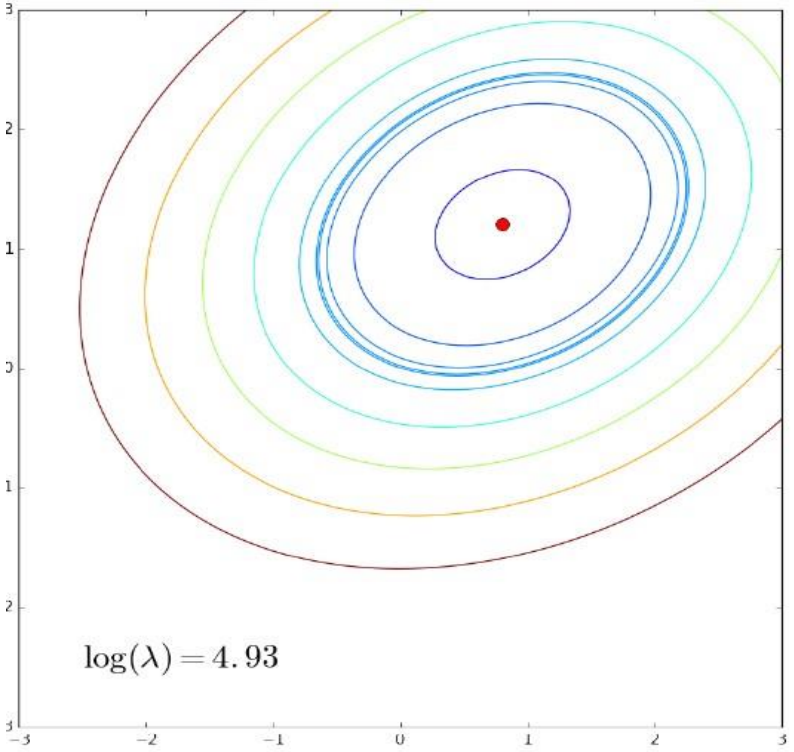
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

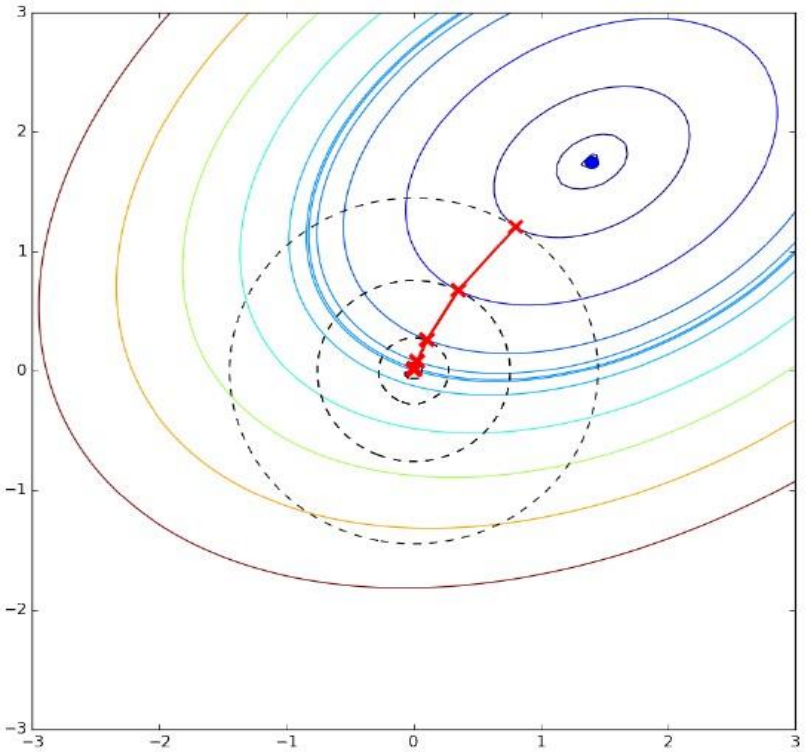
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

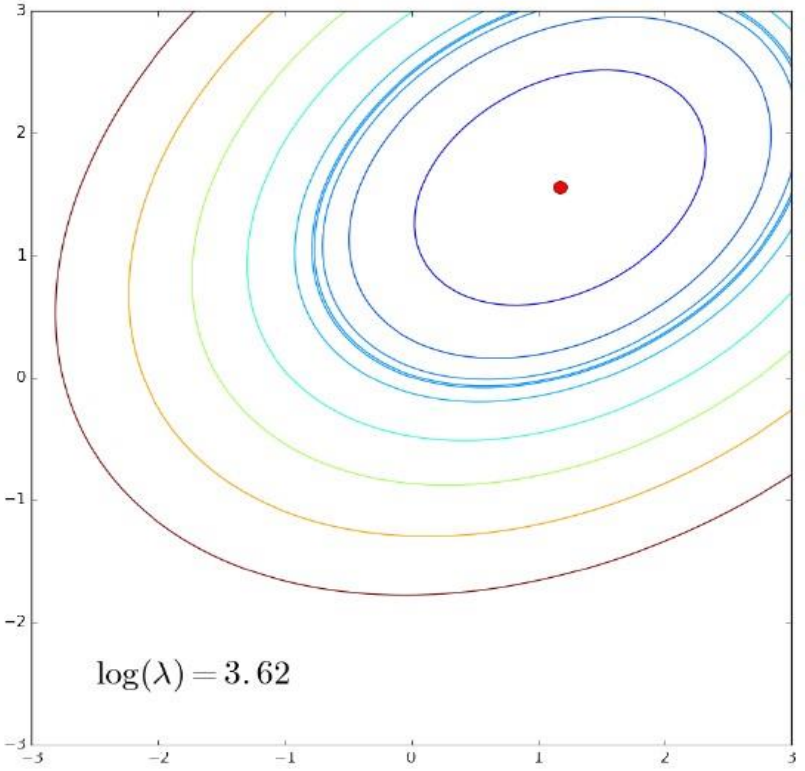
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

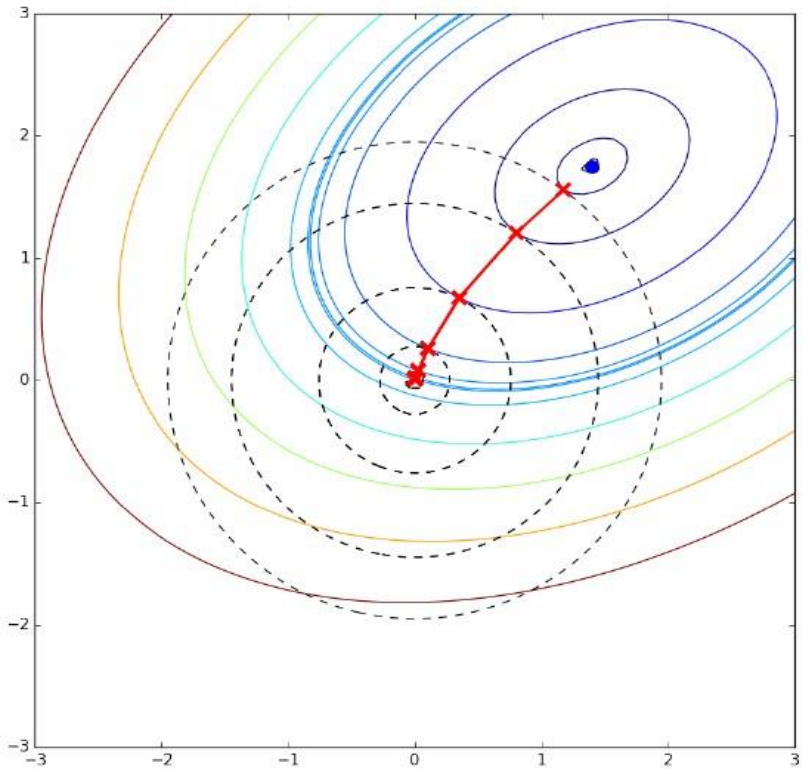
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

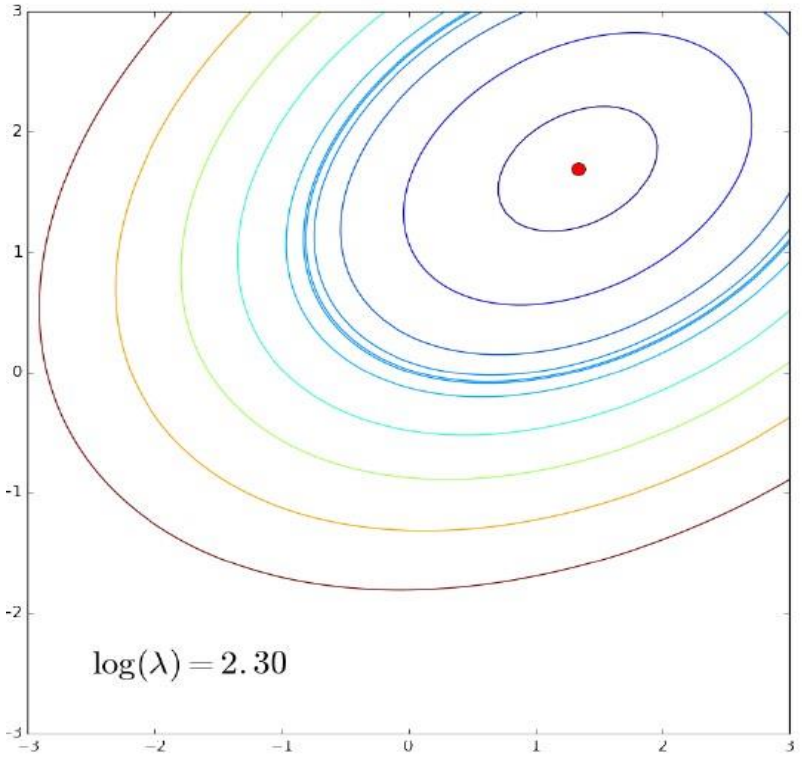
subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$



Regularization
Ridge Regression

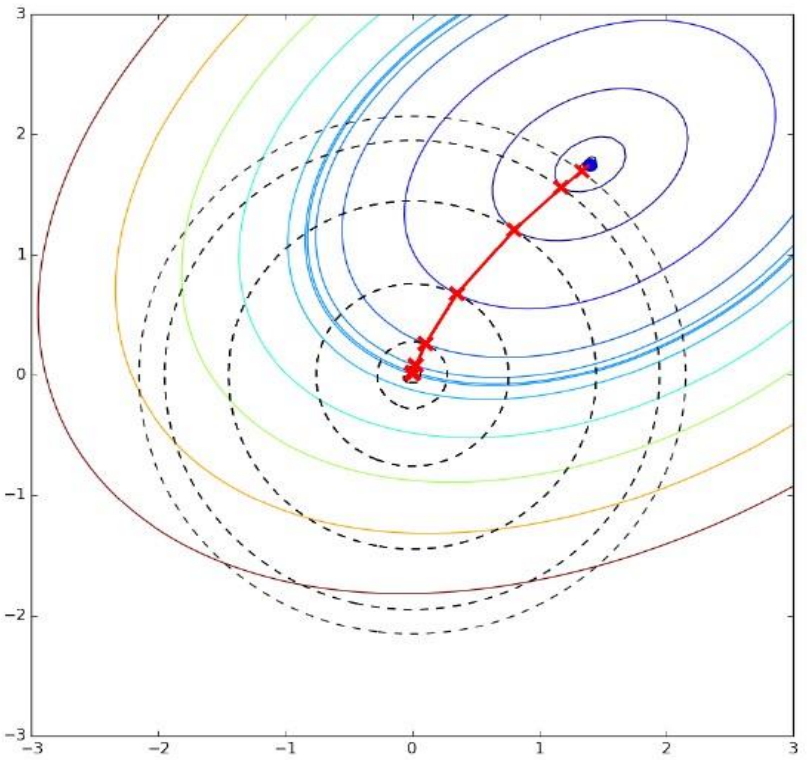
Minimise

$$(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}}\mathbf{w}$$



Minimise $(\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y})$

subject to $\mathbf{w}^{\mathsf{T}}\mathbf{w} \leq R$

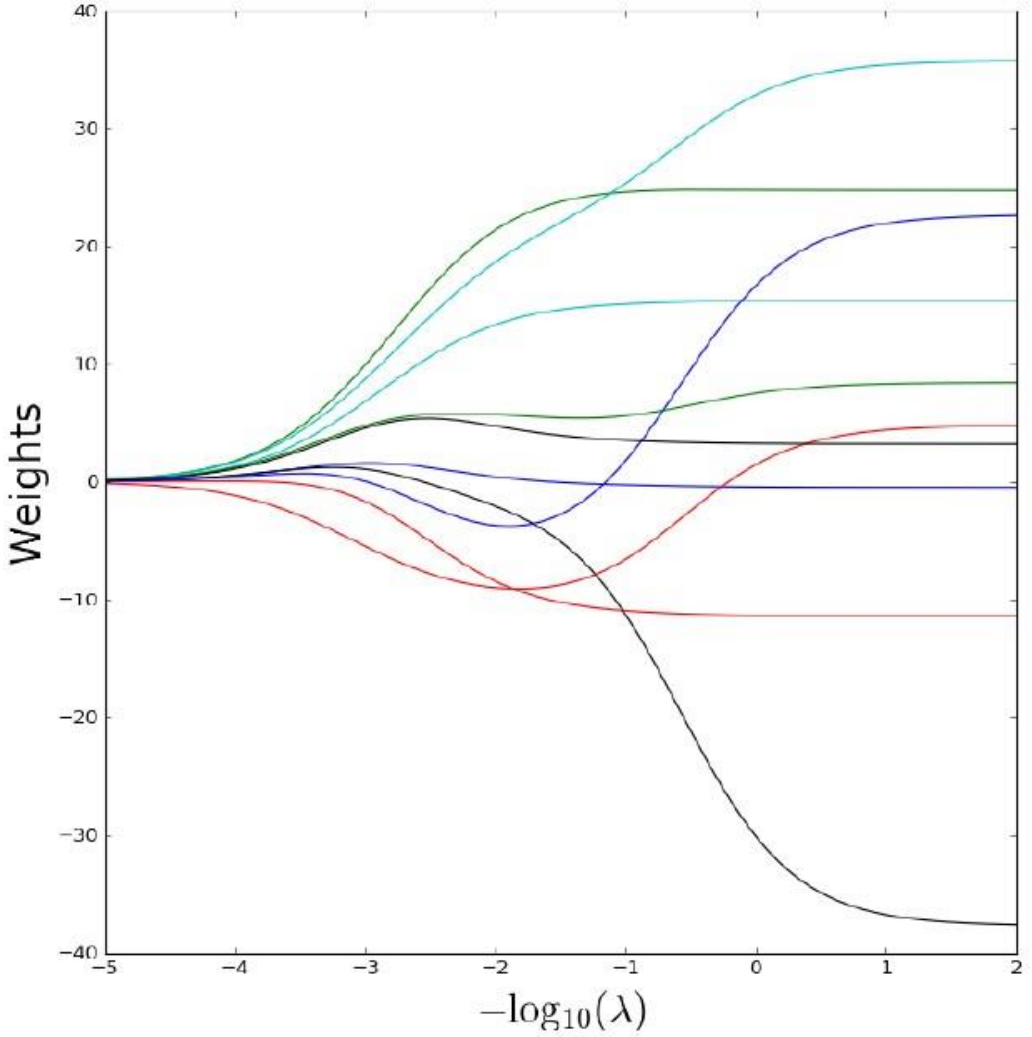


Regularization
Ridge Regression

As we decrease the magnitudes of weights start increasing

Shrinks all variables to zero – reduces variance while introducing bias.

L2 regularization promotes **grouping** – results in equal weights for correlated features



Regularization

Summary:Ridge Regression

In ridge regression, in addition to the residual sum of squares we penalise the sum of squares of weights

Ridge Regression Objective

$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^{\mathsf{T}} \mathbf{w}$$

This is also called ℓ_2-regularization or weight-decay

Penalising weights "encourages fitting signal rather than just noise"

Regularization
Lasso Regression

Lasso (least absolute shrinkage and selection operator) minimises the following objective function

Lasso Objective

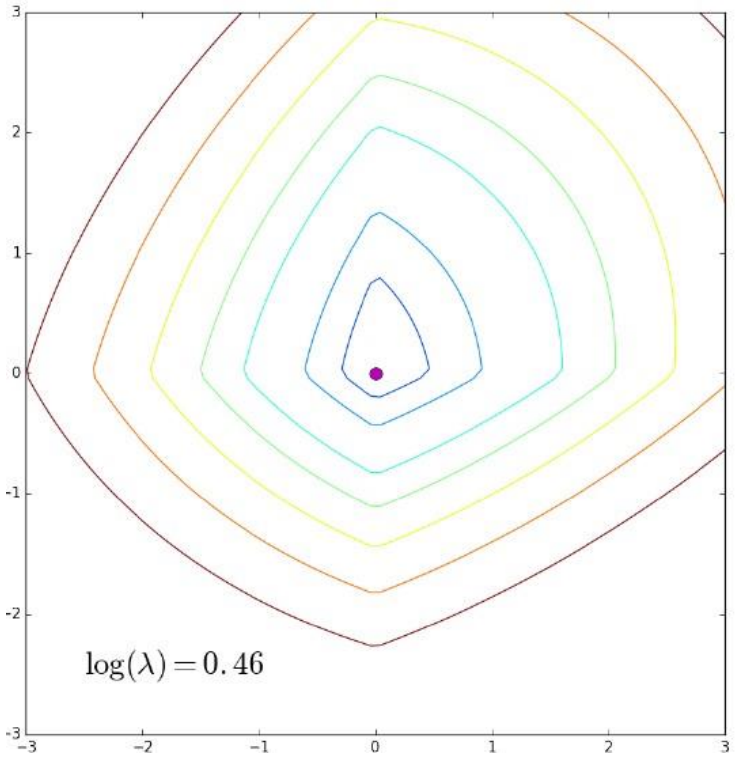
$$\mathcal{L}_{\text{lasso}}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^{\mathsf{T}} (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \sum_{i=1}^{D} |w_i|$$

- As with ridge regression, there is a penalty on the weights
- The absolute value function does not allow for a simple close-form expression (ℓ_1-regularization)
- However, there are advantages to using the lasso as we shall see next

Regularization
Lasso Regression

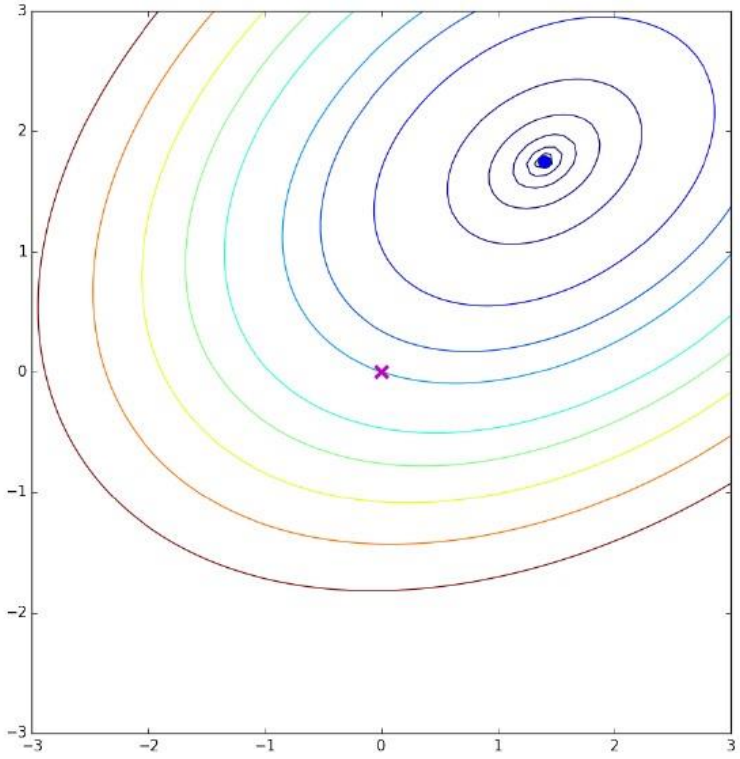
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$



Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

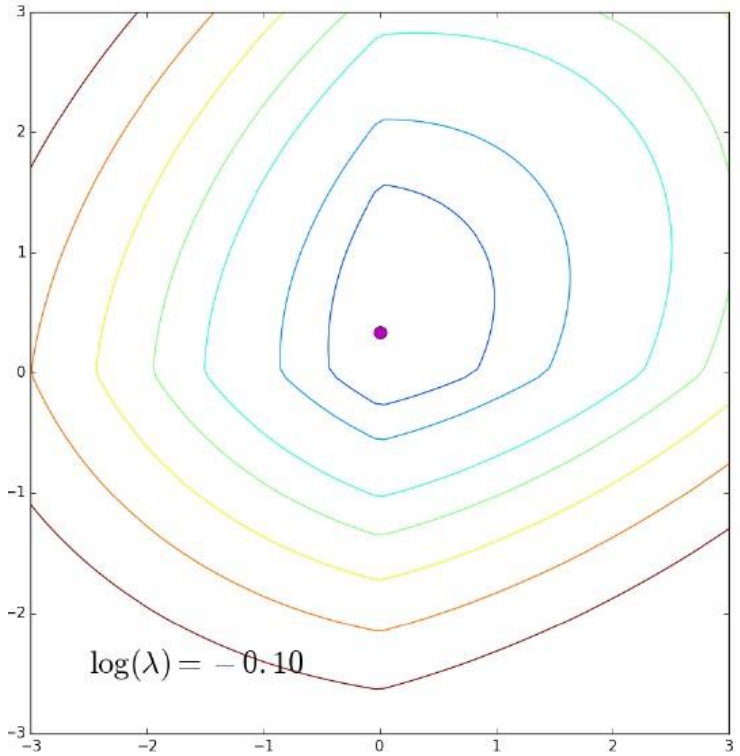
subject to $\sum_{i=1}^{D}|w_i| \leq R$



Regularization
Lasso Regression

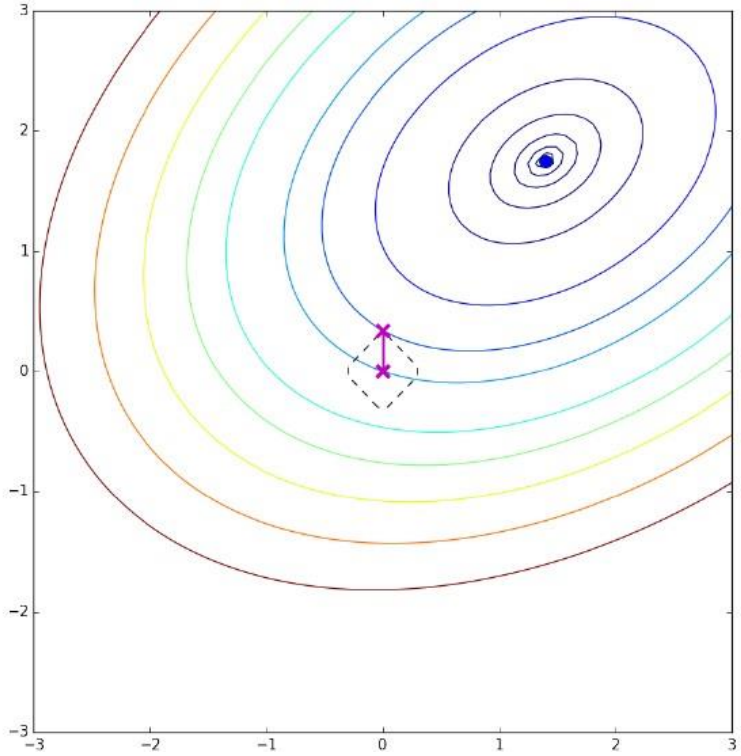
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$



Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

subject to $\sum_{i=1}^{D}|w_i| \leq R$

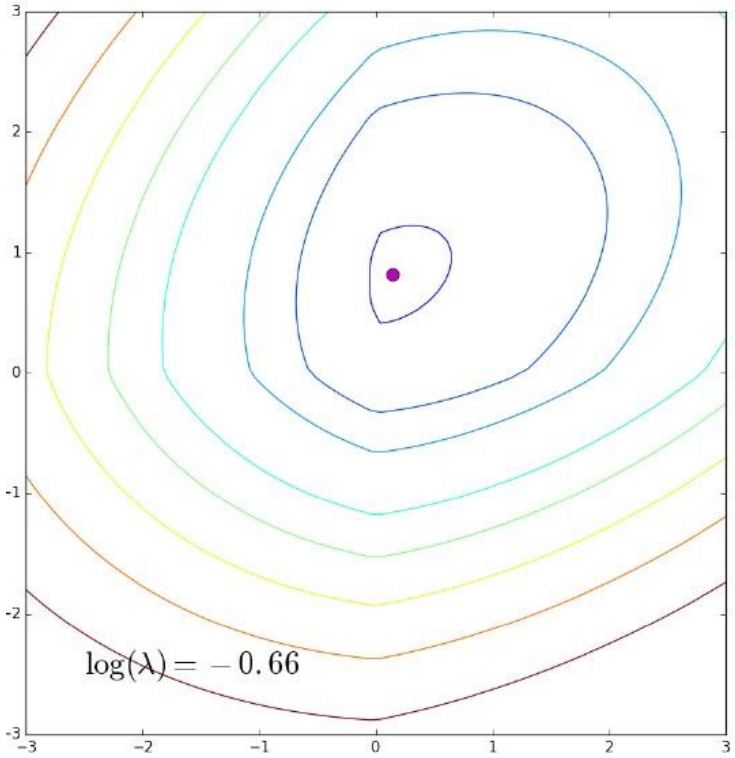


Regularization

Lasso Regression

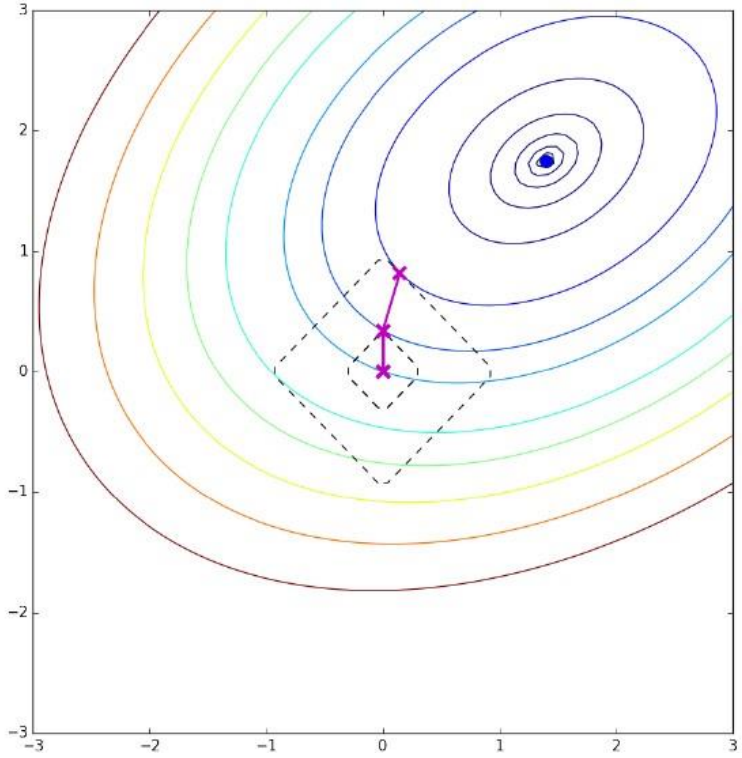
Minimise

$$(\mathbf{Xw} - \mathbf{y})^{\mathsf{T}} (\mathbf{Xw} - \mathbf{y}) + \lambda \sum_{i=1}^{D} |w_i|$$



Minimise $(\mathbf{Xw} - \mathbf{y})^{\mathsf{T}} (\mathbf{Xw} - \mathbf{y})$

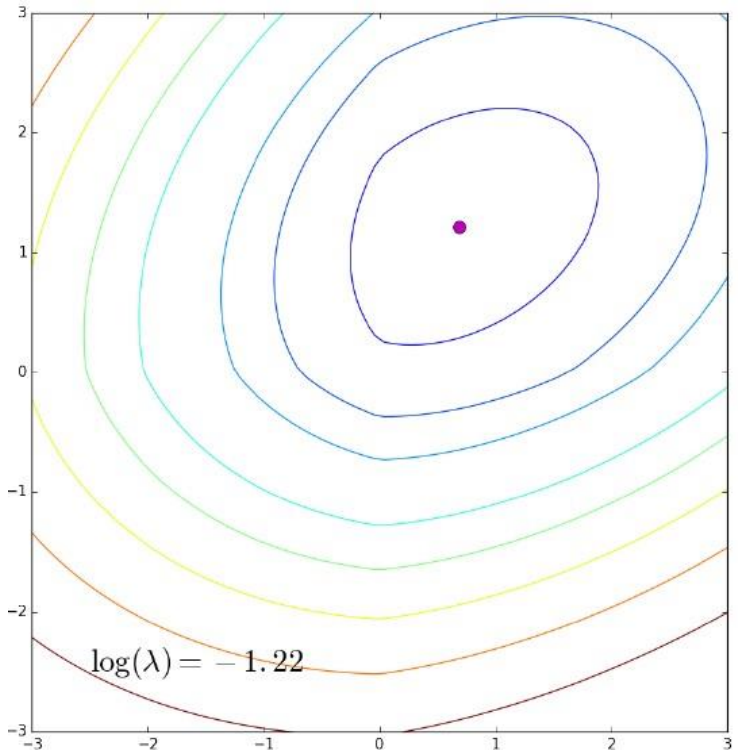
subject to $\sum_{i=1}^{D} |w_i| \leq R$



Regularization
Lasso Regression

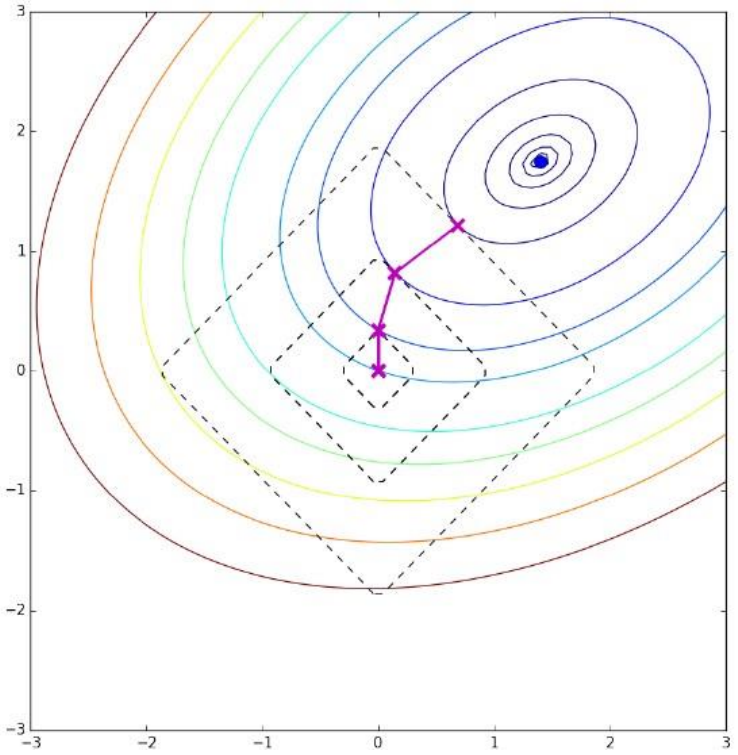
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$



Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

subject to $\sum_{i=1}^{D}|w_i| \leq R$



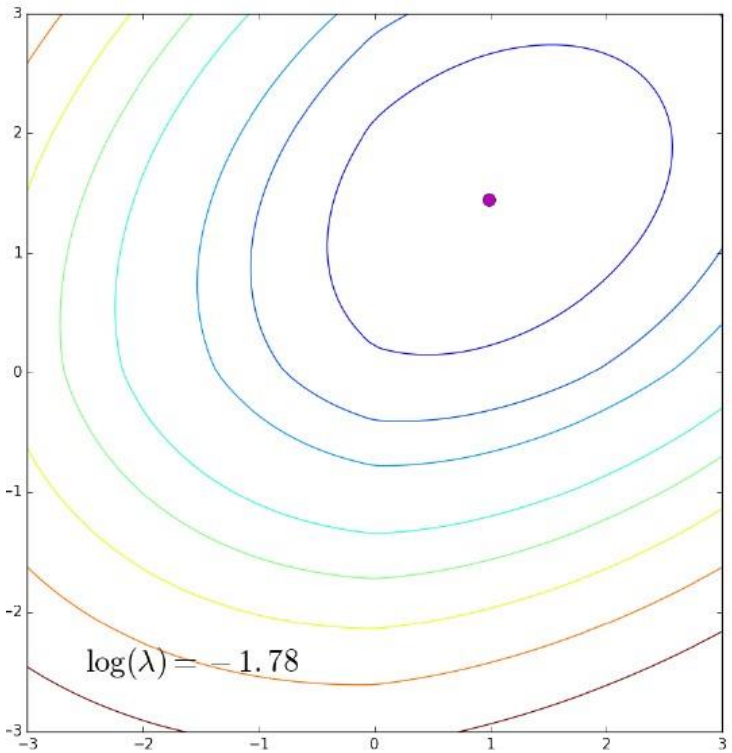
Regularization
Lasso Regression

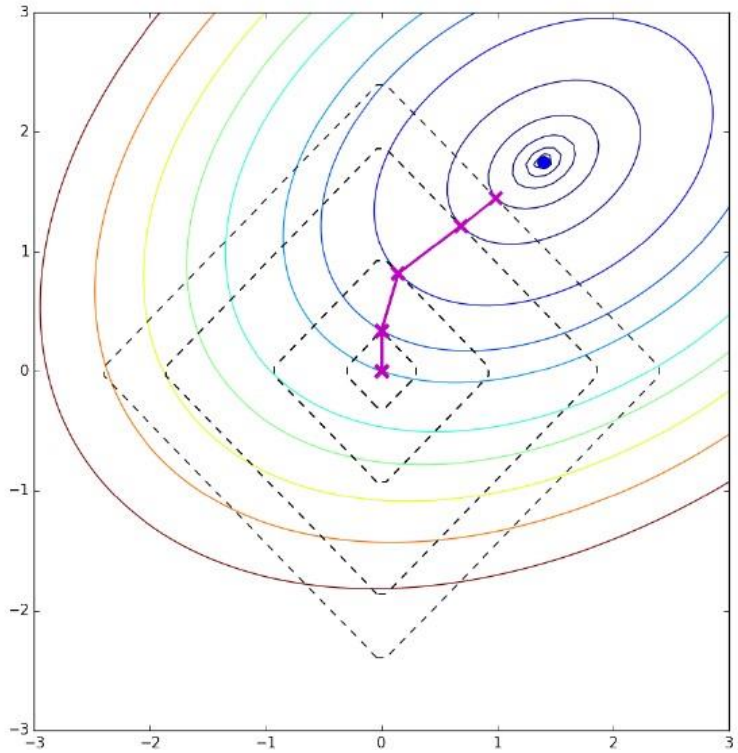
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$

Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

subject to $\sum_{i=1}^{D}|w_i|\leq R$

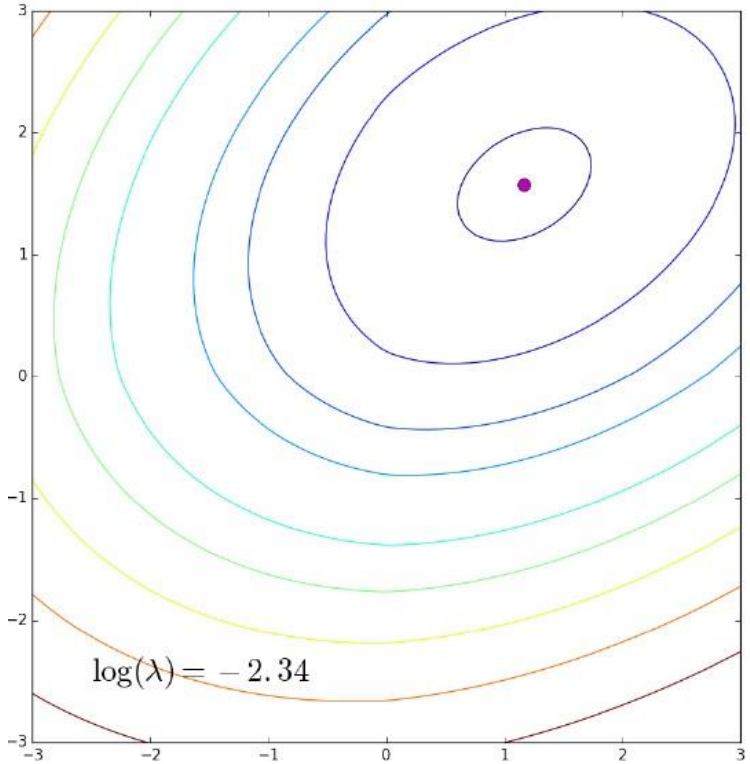




Regularization
Lasso Regression

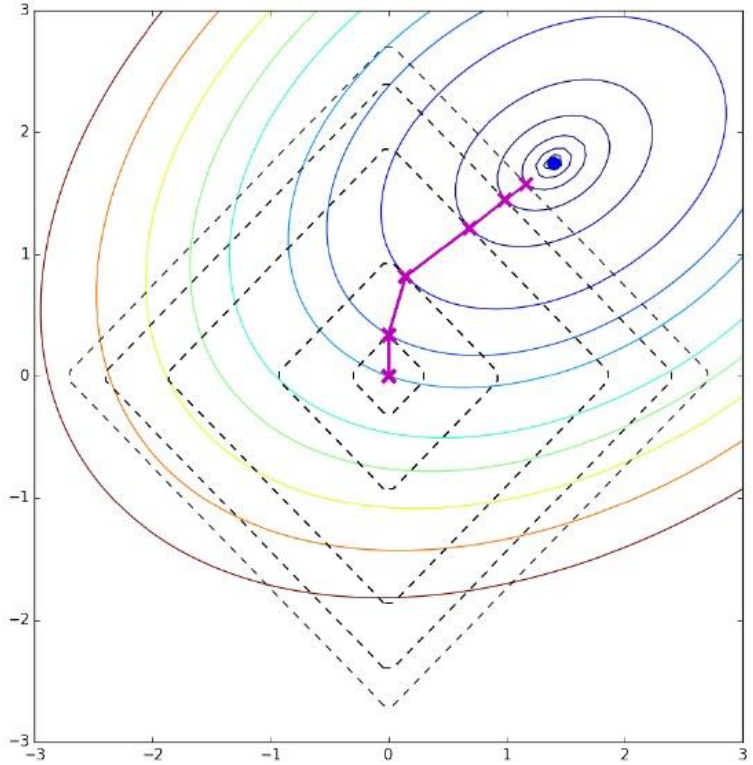
Minimise

$$(\mathbf{Xw} - \mathbf{y})^{\mathsf{T}} (\mathbf{Xw} - \mathbf{y}) + \lambda \sum_{i=1}^{D} |w_i|$$



Minimise $(\mathbf{Xw} - \mathbf{y})^{\mathsf{T}} (\mathbf{Xw} - \mathbf{y})$

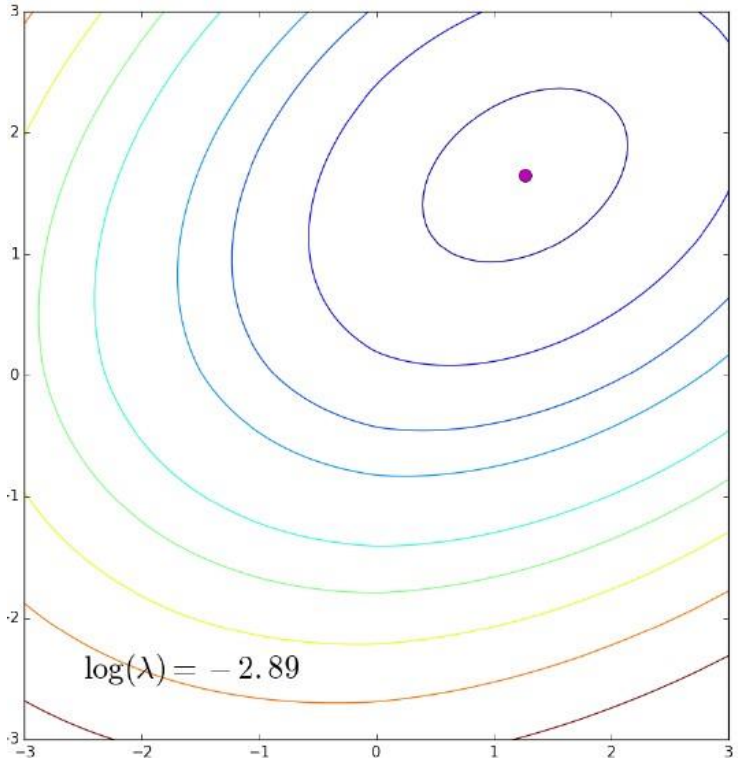
subject to $\sum_{i=1}^{D} |w_i| \leq R$



Regularization
Lasso Regression

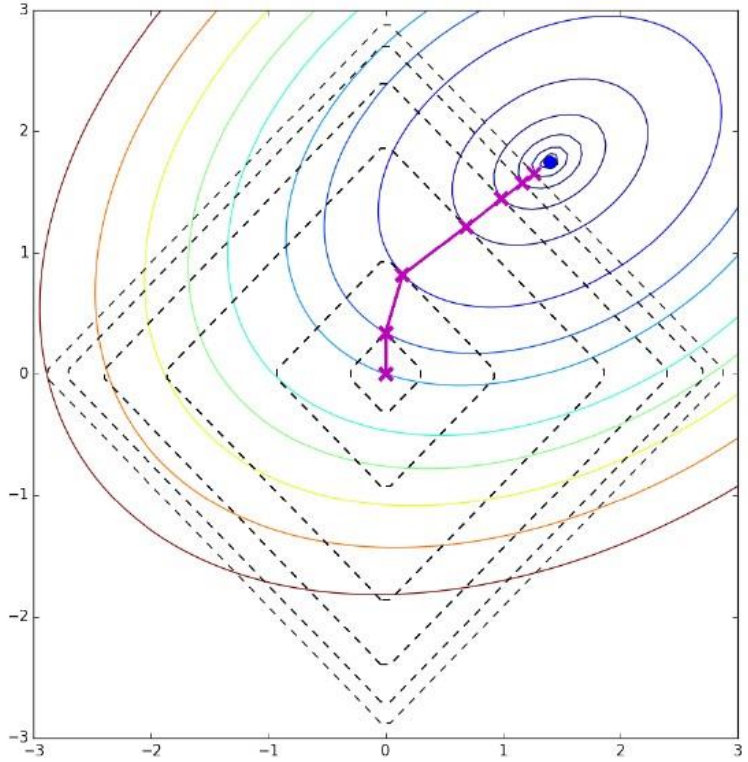
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$



Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

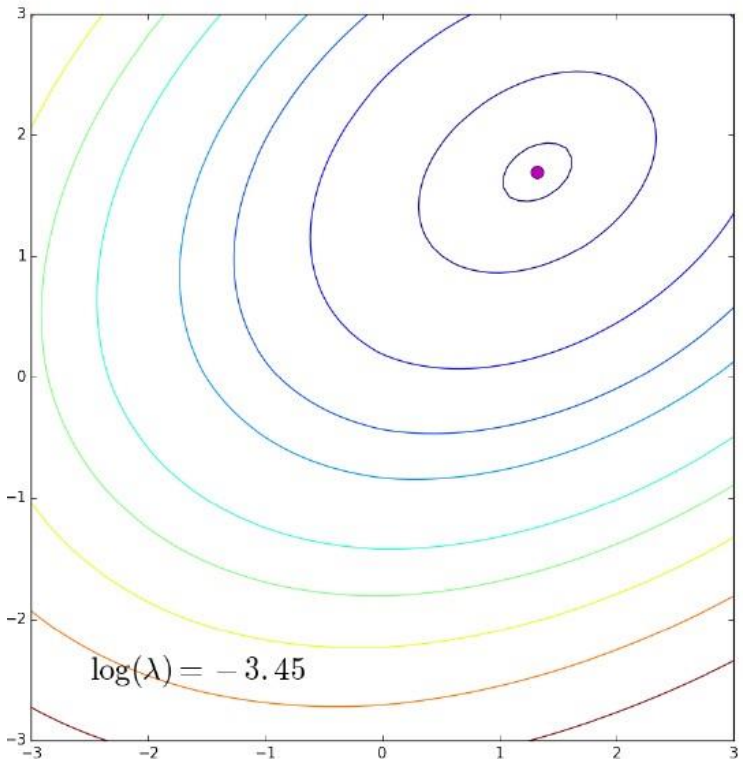
subject to $\sum_{i=1}^{D}|w_i| \leq R$



Regularization
Lasso Regression

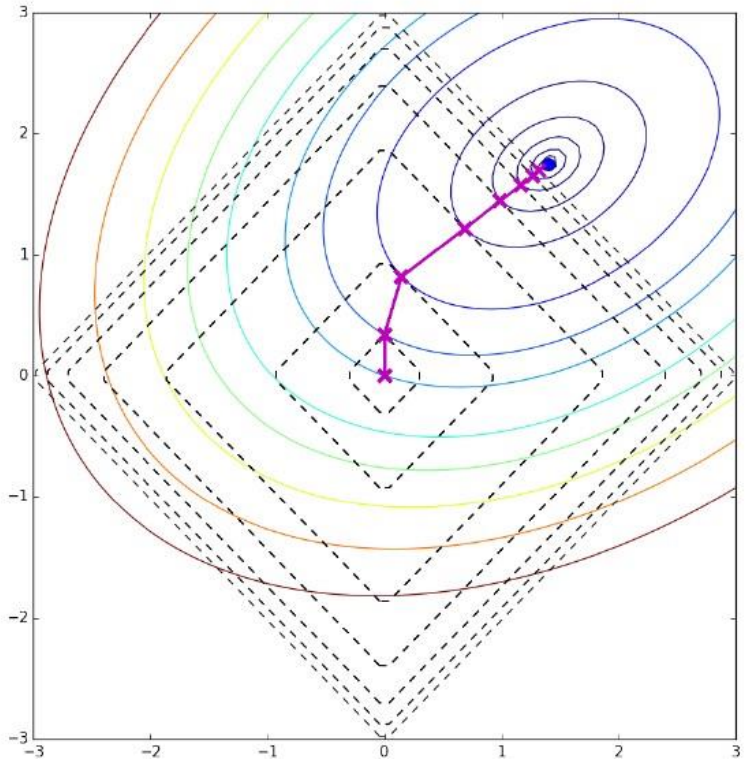
Minimise

$$(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})+\lambda\sum_{i=1}^{D}|w_i|$$



Minimise $(\mathbf{Xw}-\mathbf{y})^{\mathsf{T}}(\mathbf{Xw}-\mathbf{y})$

subject to $\sum_{i=1}^{D}|w_i| \leq R$

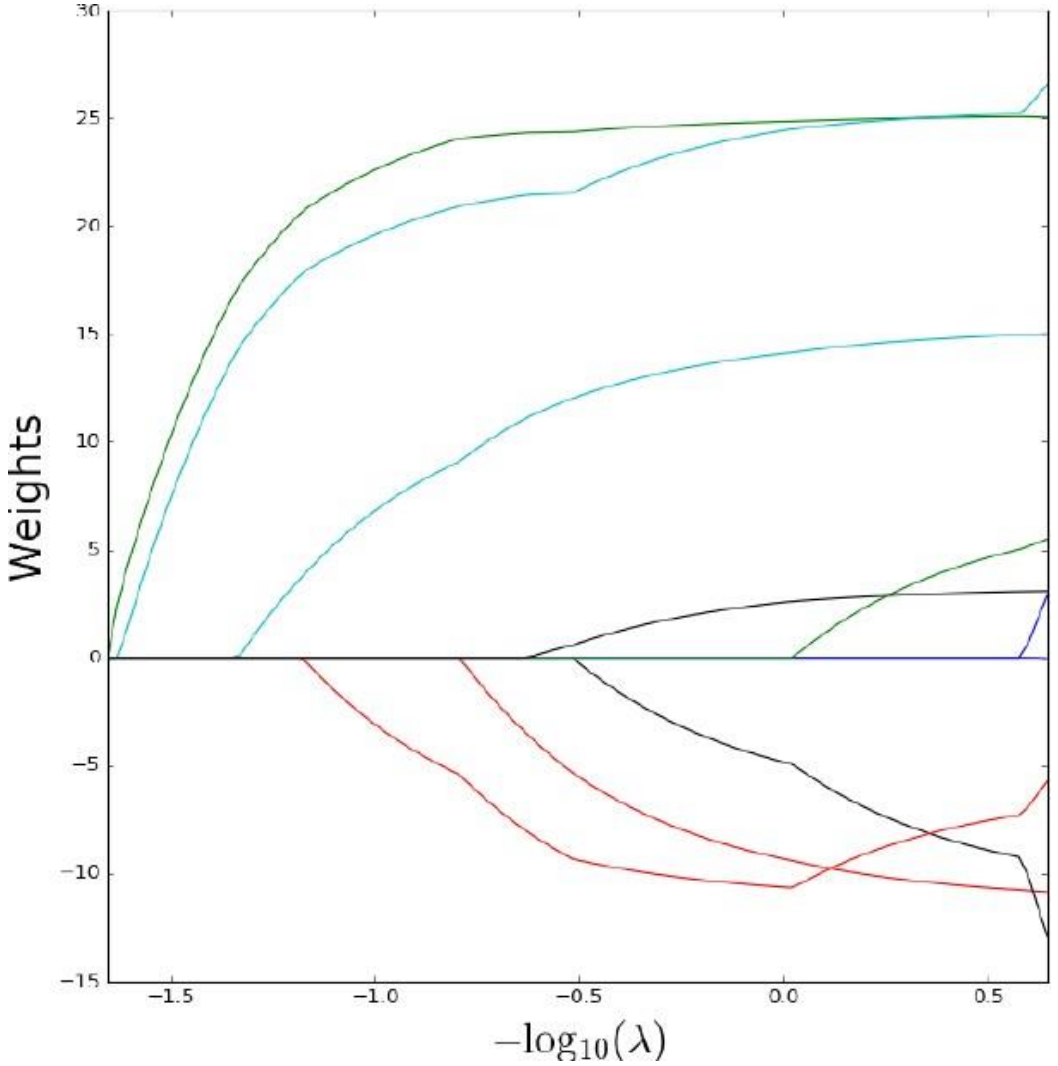


Regularization
Lasso Regression

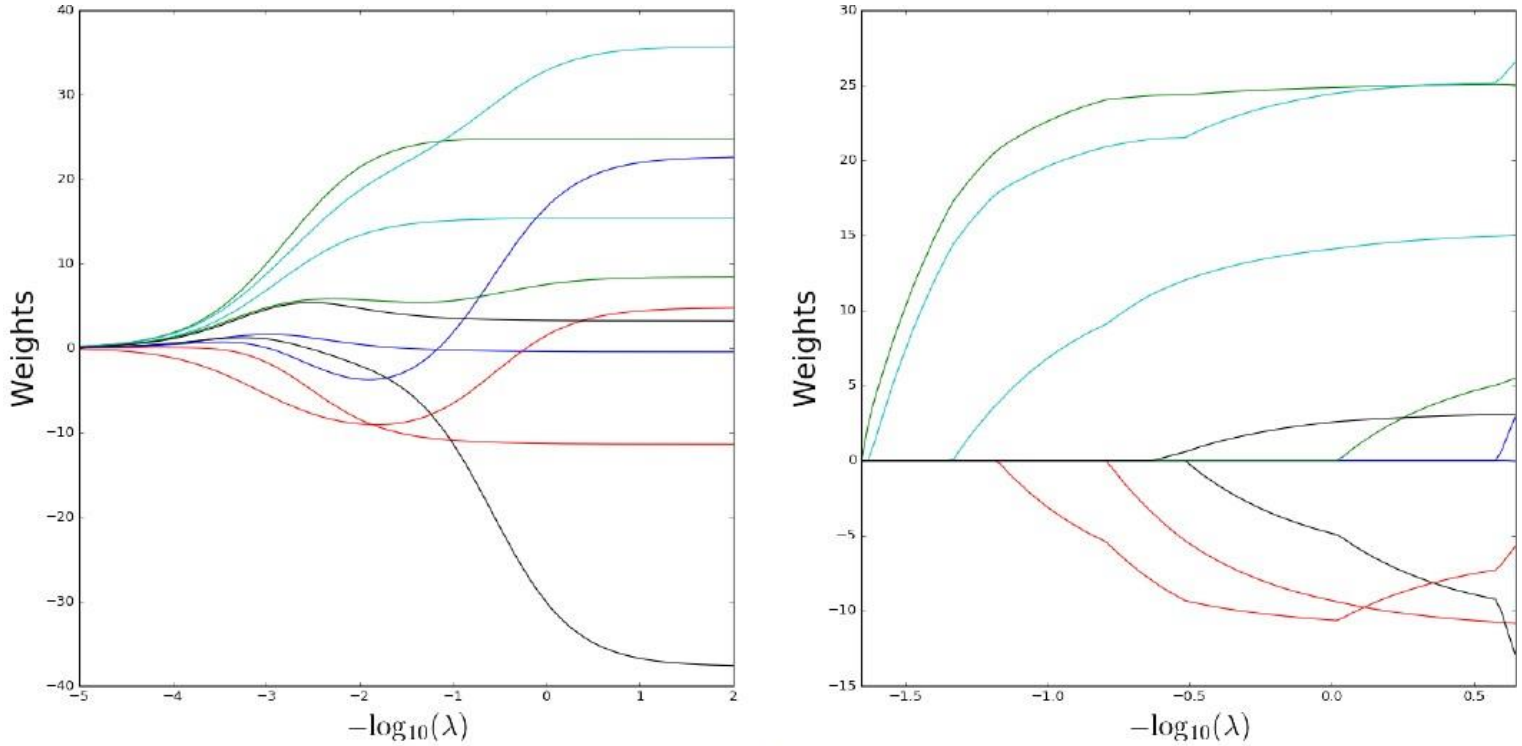
As we decrease the magnitudes of weights start increasing

Not all variables become zero at the same time – results in variable selection

L1 regularization promotes **sparsity** – selects few informative features



Regularization
Lasso Regression



When using the Lasso, weights are often exactly 0.

Thus, Lasso gives sparse models.

Regularization

L2 vs L1Regularization

Example: Assume 3 input features: $\mathbf{x} = (1, 2, 1)^\top$

The following two **linear classifiers** $f_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$ yield the **same result/loss:**

- $\mathbf{w}_1 = (0, 0.75, 0)^\top \Rightarrow$ ignores 2 features
- $\mathbf{w}_2 = (0.25, 0.5, 0.25)^\top \Rightarrow$ takes all features into account

But the L1 and L2 regularizer **prefer different solutions!**

L2 Regularization:

- $\|\mathbf{w}_1\|_2 = 0 + 0.75^2 + 0 = 0.5625$
- $\|\mathbf{w}_2\|_2 = 0.25^2 + 0.5^2 + 0.25^2 = \mathbf{0.375}$

L1 Regularization:

- $\|\mathbf{w}_1\|_1 = 0 + 0.75 + 0 = \mathbf{0.75}$
- $\|\mathbf{w}_2\|_1 = 0.25 + 0.5 + 0.25 = 1$

Regularization
Summary and relation to NNs

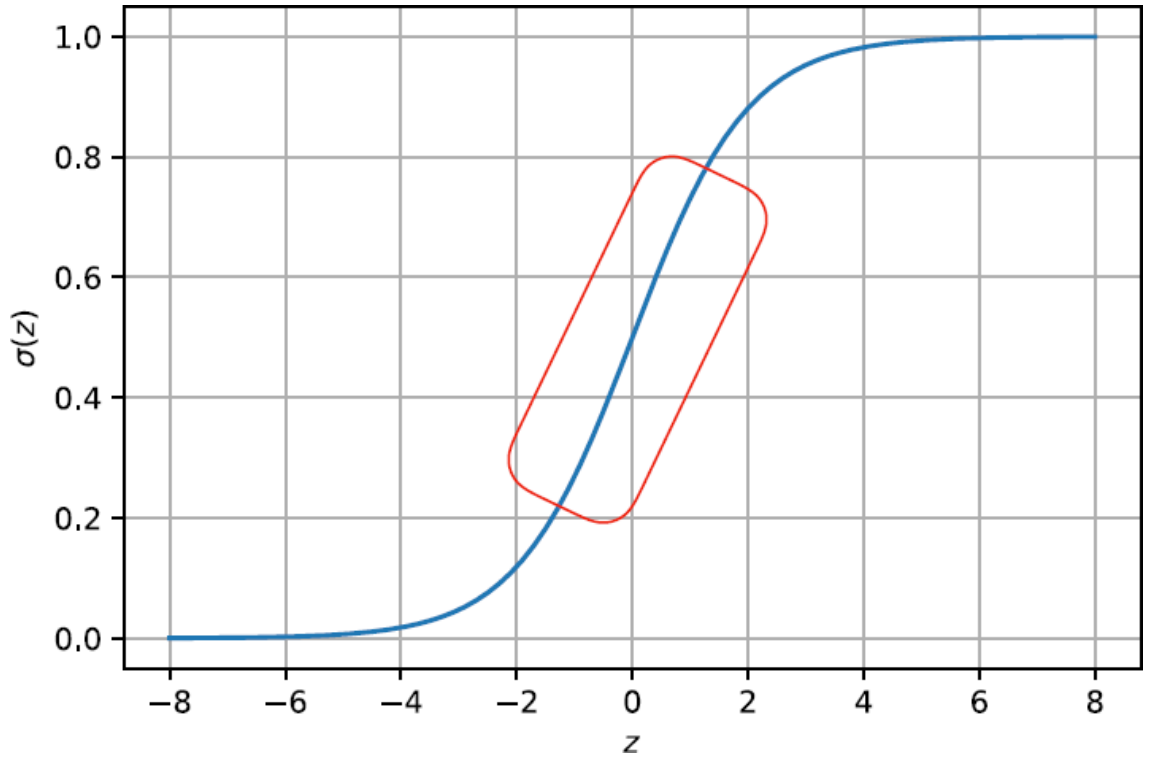
Smaller weights: Early stopping, norm penalties

Now: Other Ways for Dealing with Overfitting if Collecting More Data is not Feasible →
Reducing Network's Capacity by Other Means

Smaller architecture:

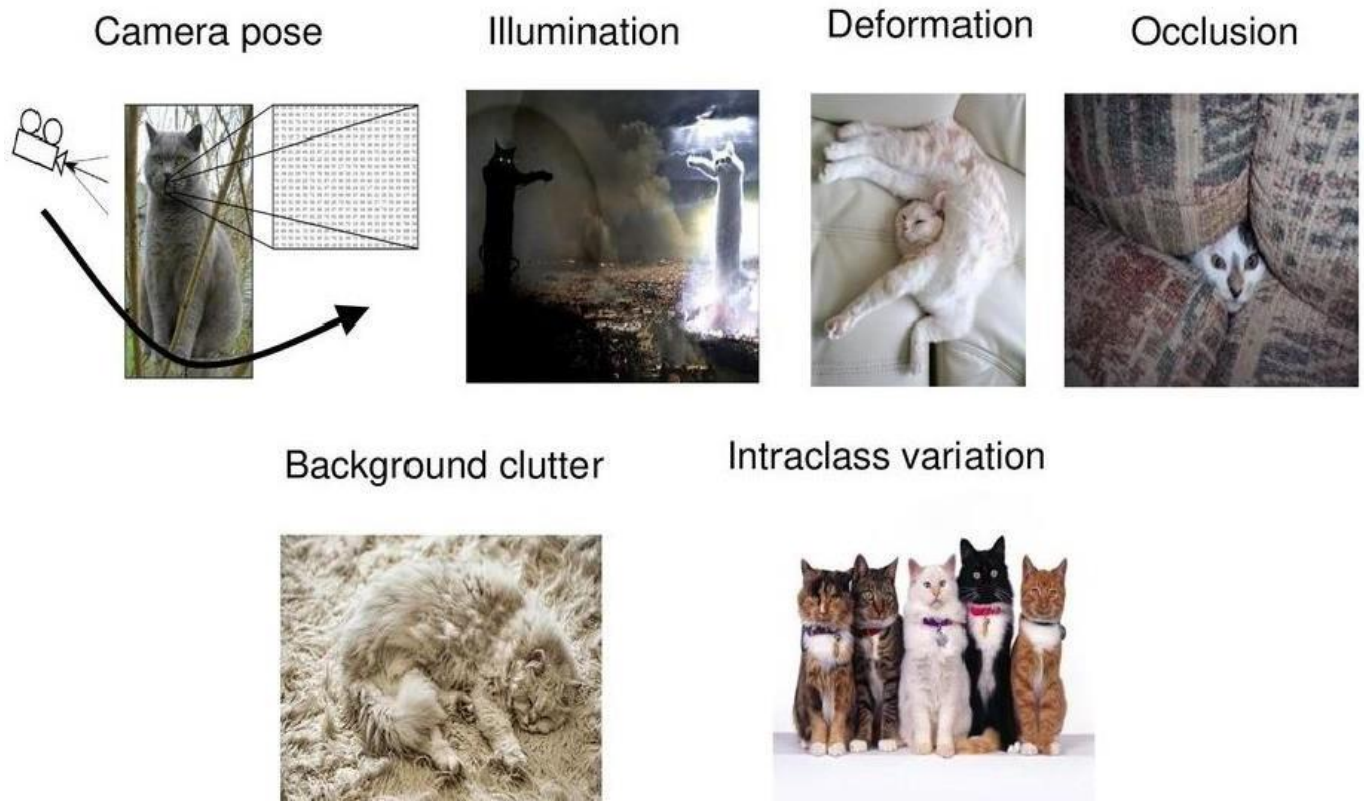
Fewer hidden layers & units
Dropout
Dead ReLUs
L1 norm penalty

Adding noise: Dropout



Data augmentation
Invariance Problem

An often-repeated claim about CNNs is that they are invariant to small translations. Independently of whether this is true, they are not invariant to most other types of transformations:



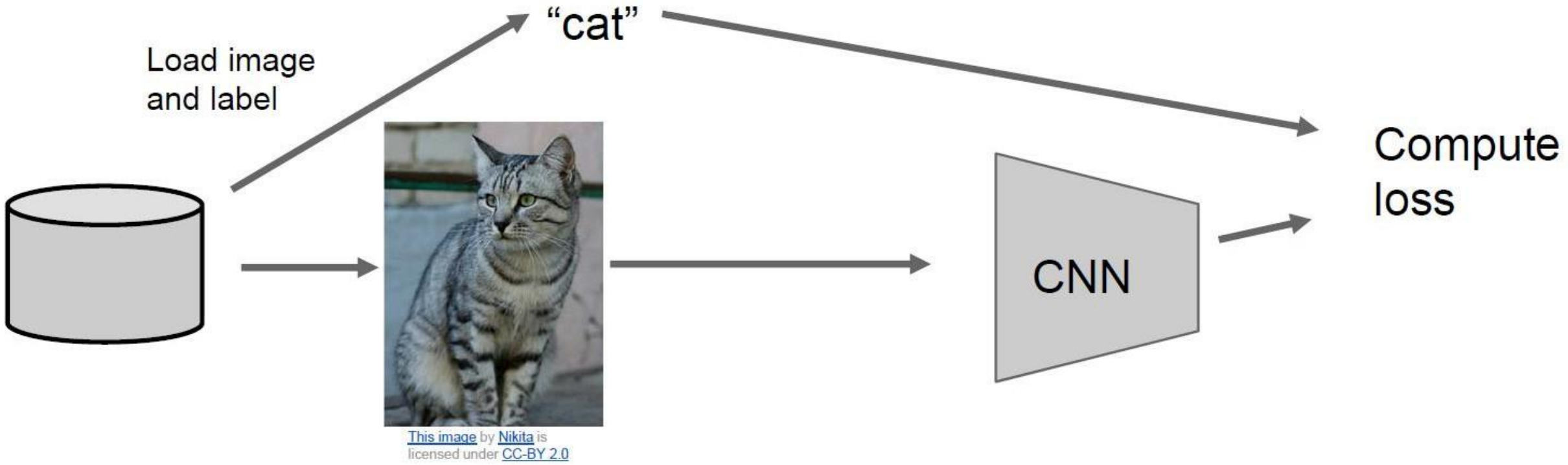
Data augmentation

Invariance Problem

- Can greatly increase the amount of data by performing:
  - Translations
  - Rotations
  - Reflections
  - Scaling
  - Cropping
  - Adding Gaussian Noise
  - Adding Occlusion
  - Interpolation
  - etc.
- Crucial for achieving state-of-the-art performance!
- For example, ResNet improves from 11.66% to 6.41% error on CIFAR-10 dataset and from 44.74% to 27.22% on CIFAR-100.

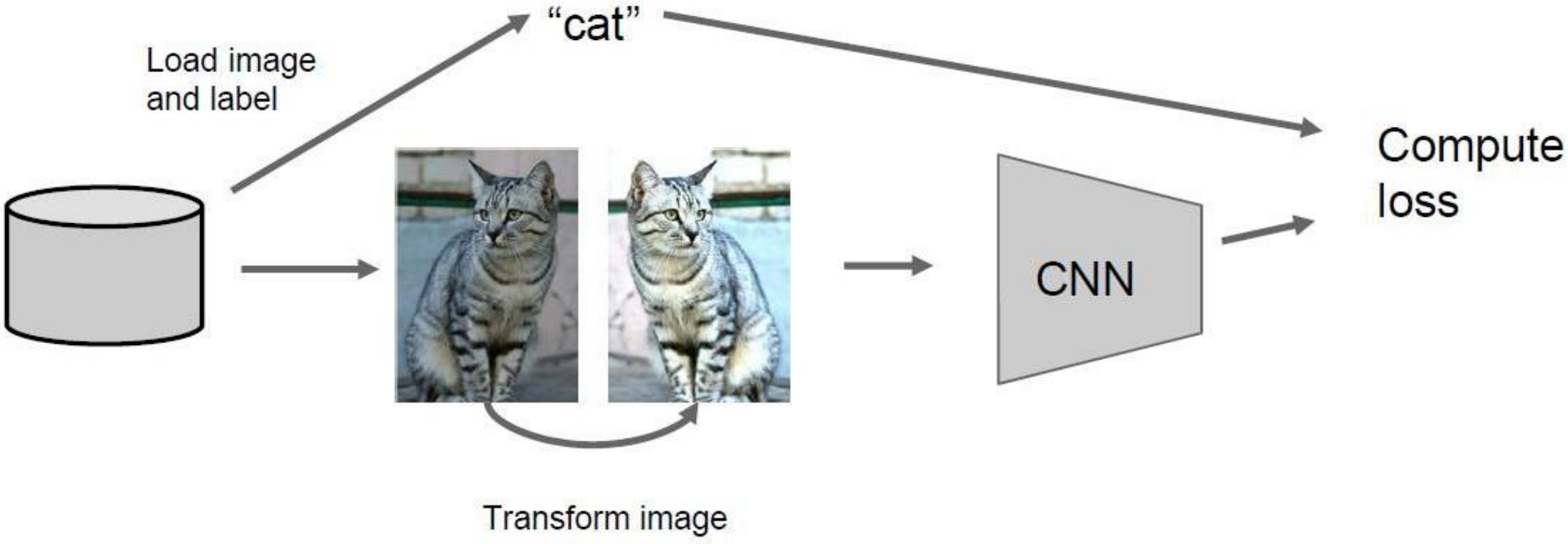
Data augmentation

Invariance Problem



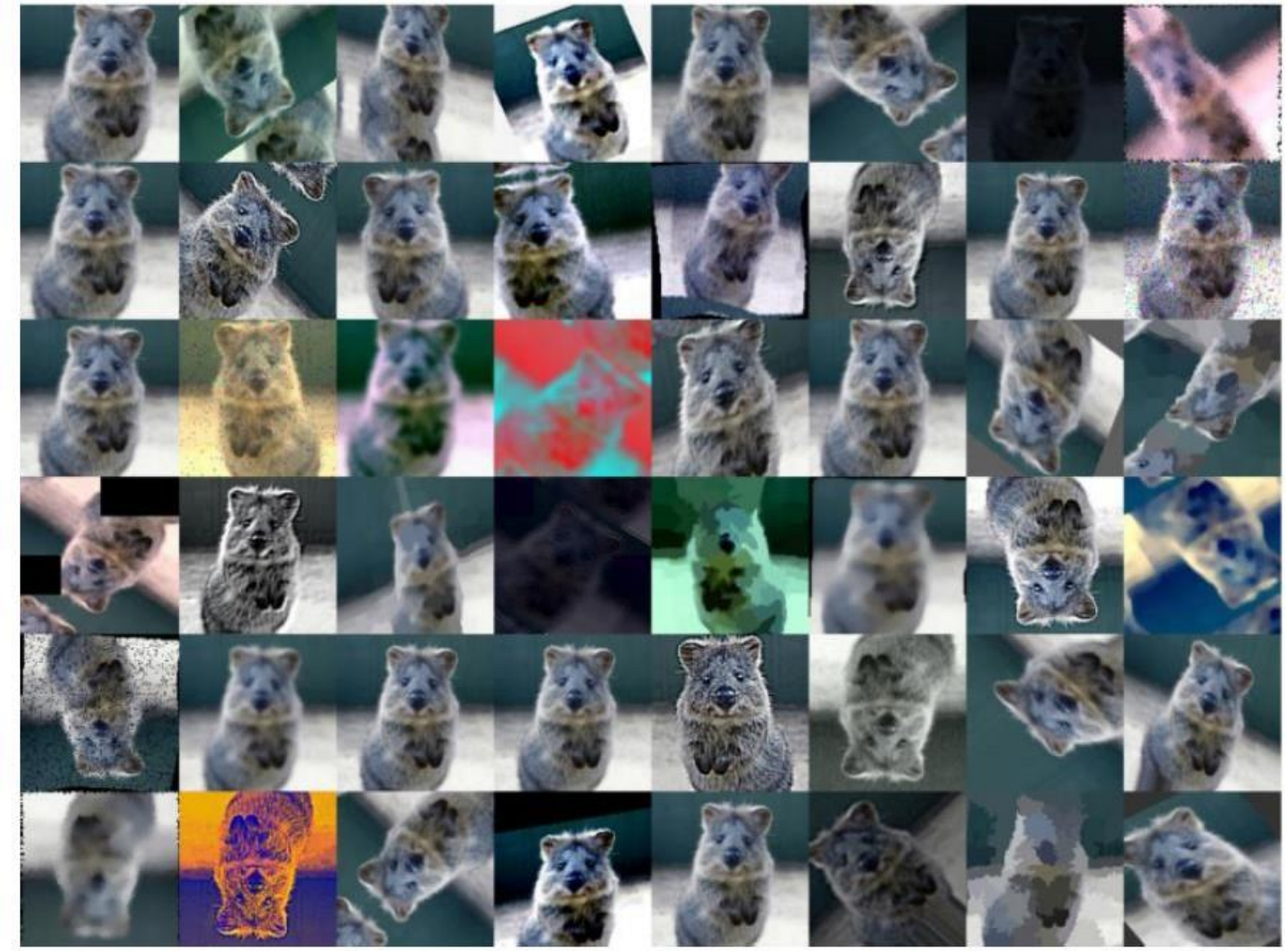
Data augmentation

Invariance Problem

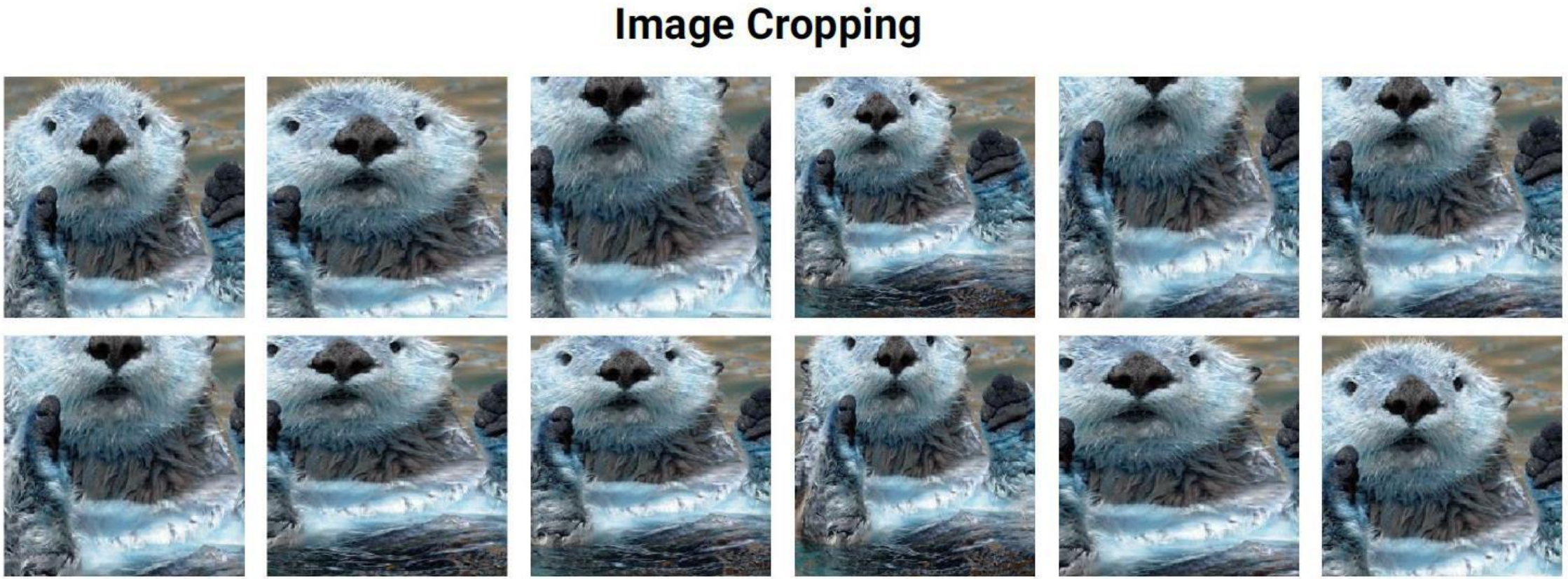


Data augmentation

Invariance Problem



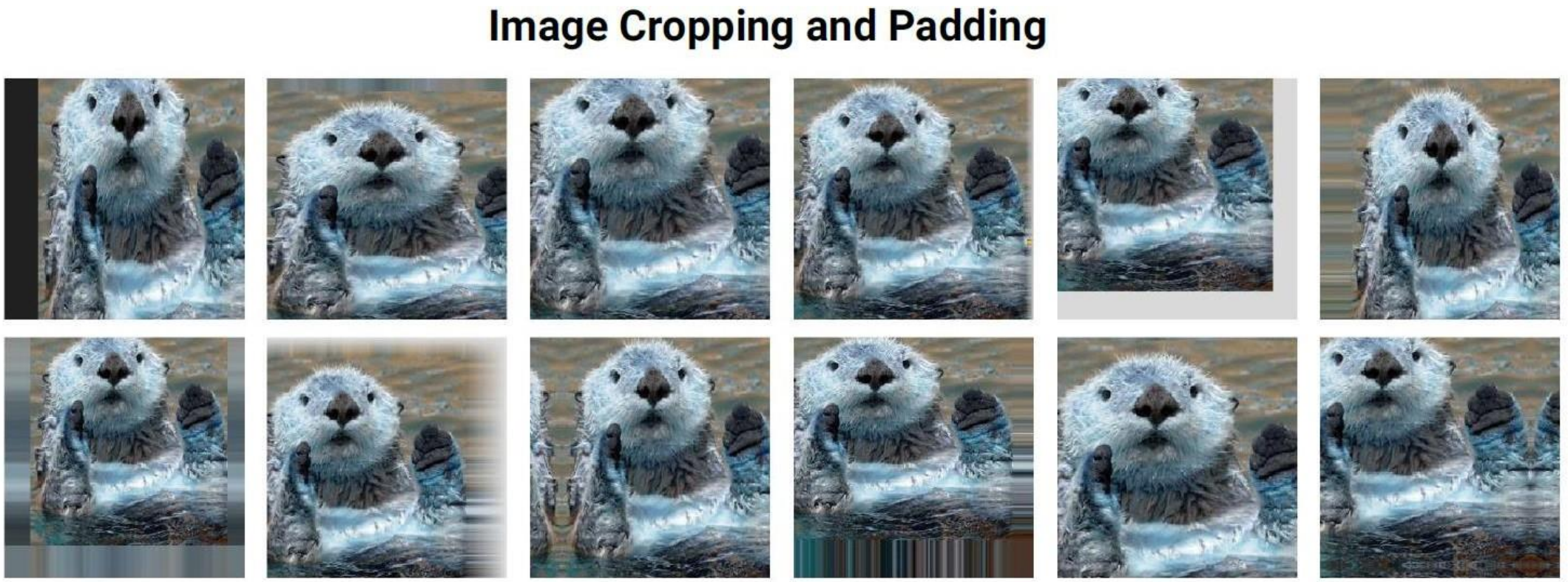
Data augmentation
Geometric Transformations



`iaa.Crop(px=(1,64))`

Data augmentation

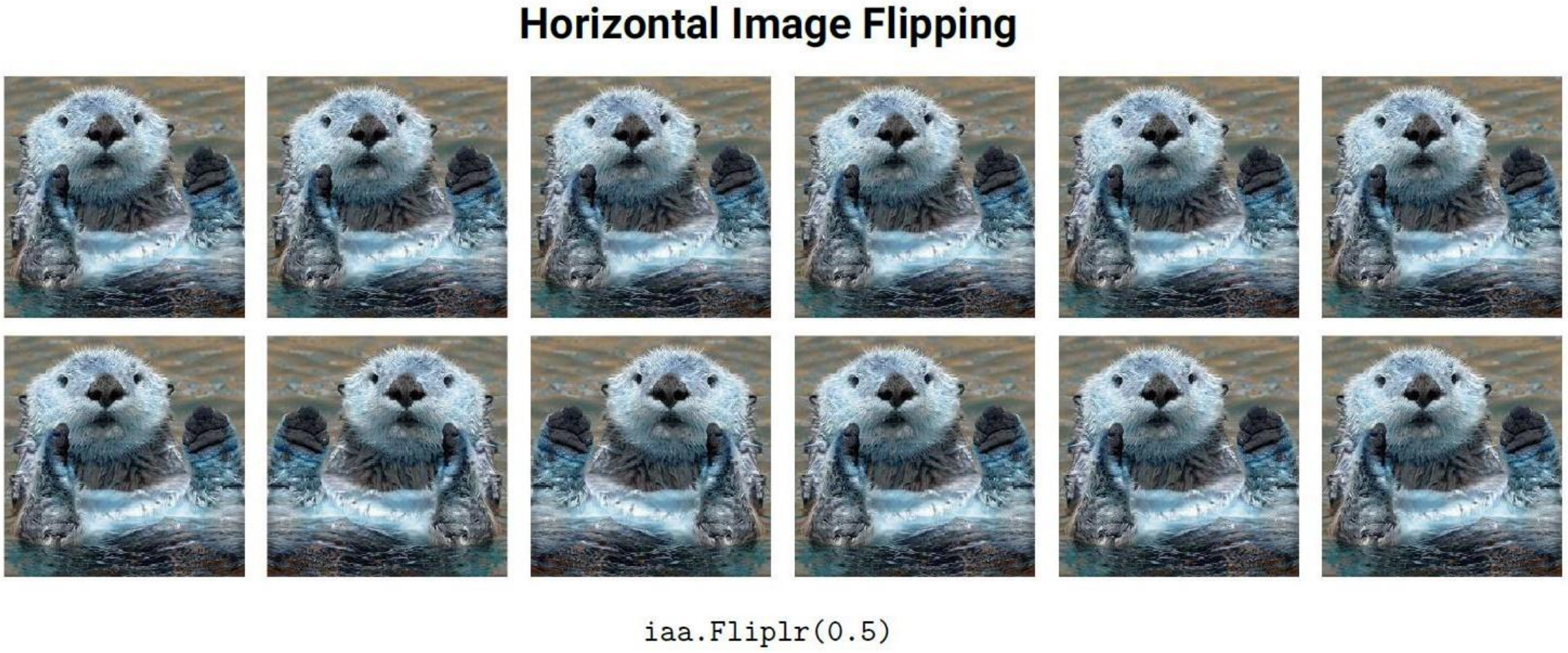
Geometric Transformations




```
iaa.CropAndPad(percent=(-0.2, 0.2),pad_mode=iaa.ia.ALL,pad_cval=(0, 255))
```

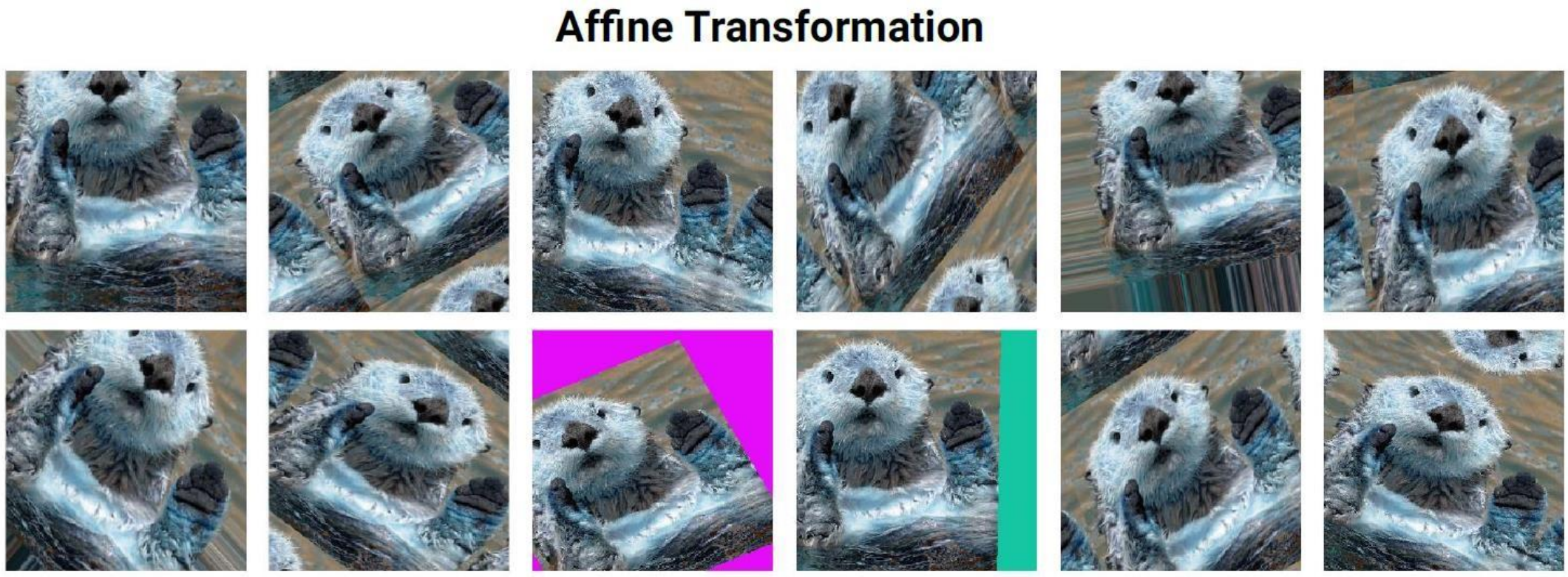
Data augmentation

Geometric Transformations



Data augmentation

Geometric Transformations



`iaa.Affine()`

Data augmentation
Geometric Transformations

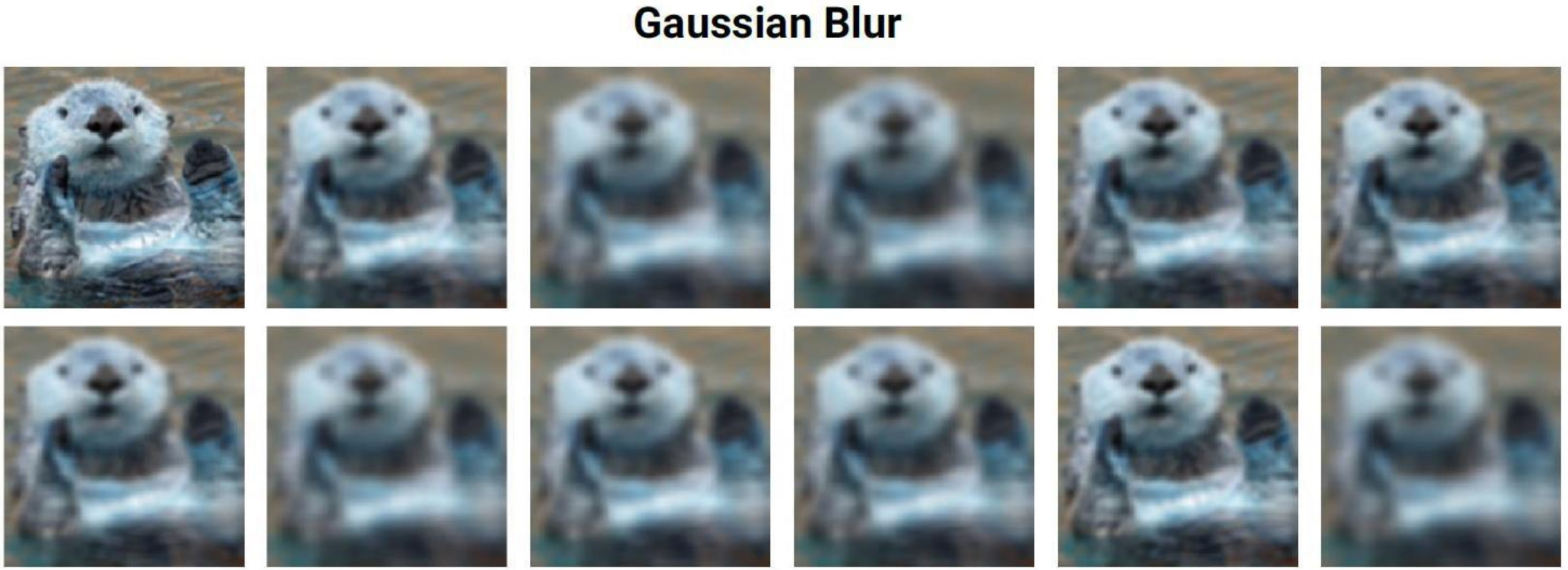

```
iaa.PiecewiseAffine(scale=(0.01, 0.1))
```

Data augmentation

Geometric Transformations

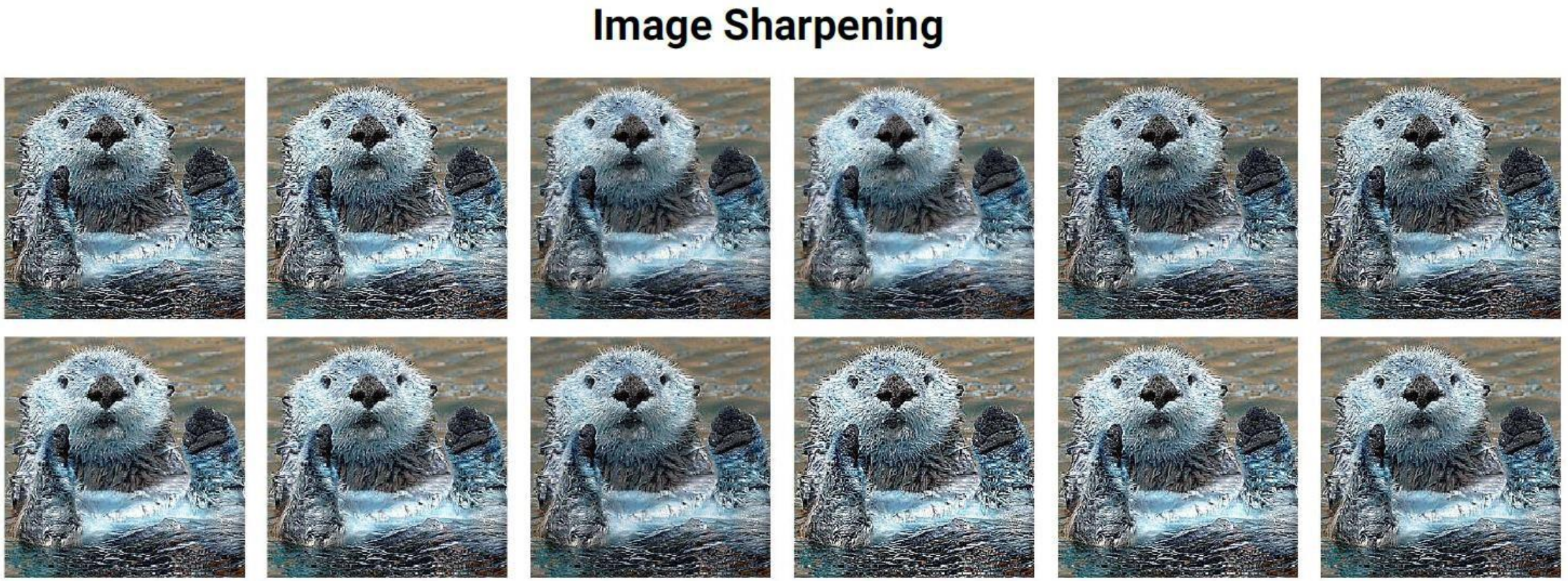

```
iaa.PerspectiveTransform(scale=(0, 0.4))
```

Data augmentation
Local Filters



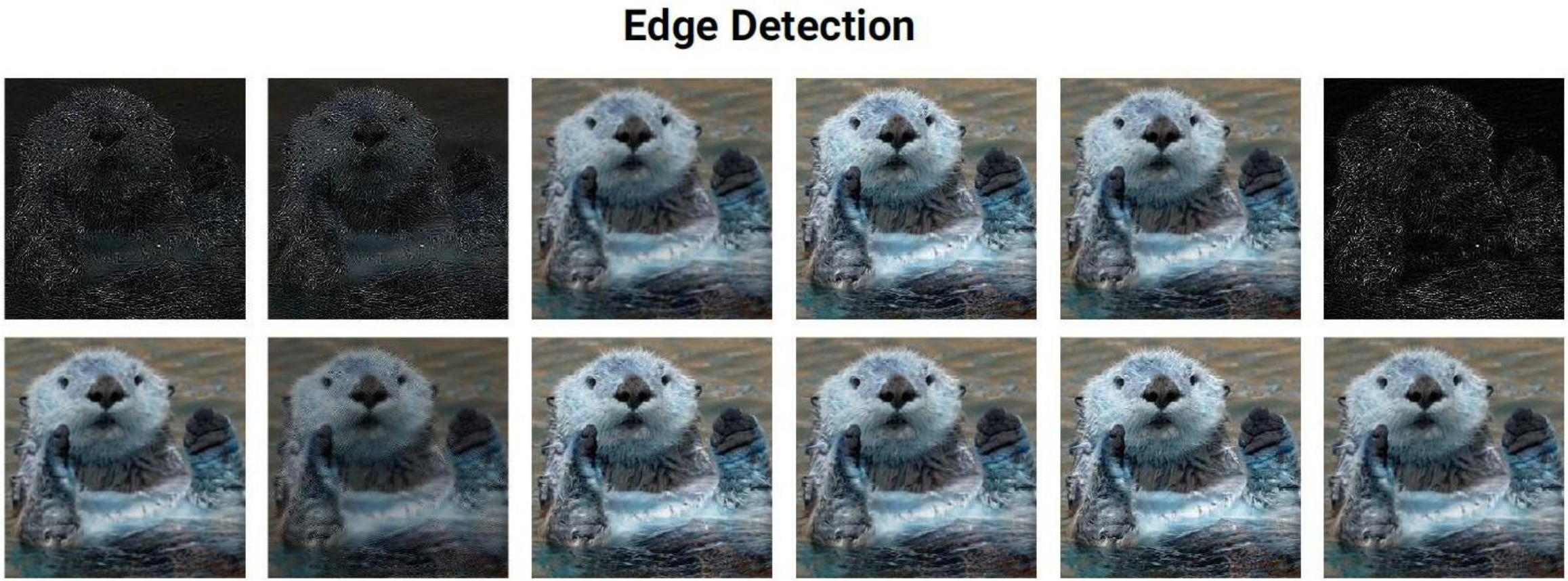

```
iaa.GaussianBlur(sigma=(0.0, 10.0))
```

Data augmentation
Local Filters



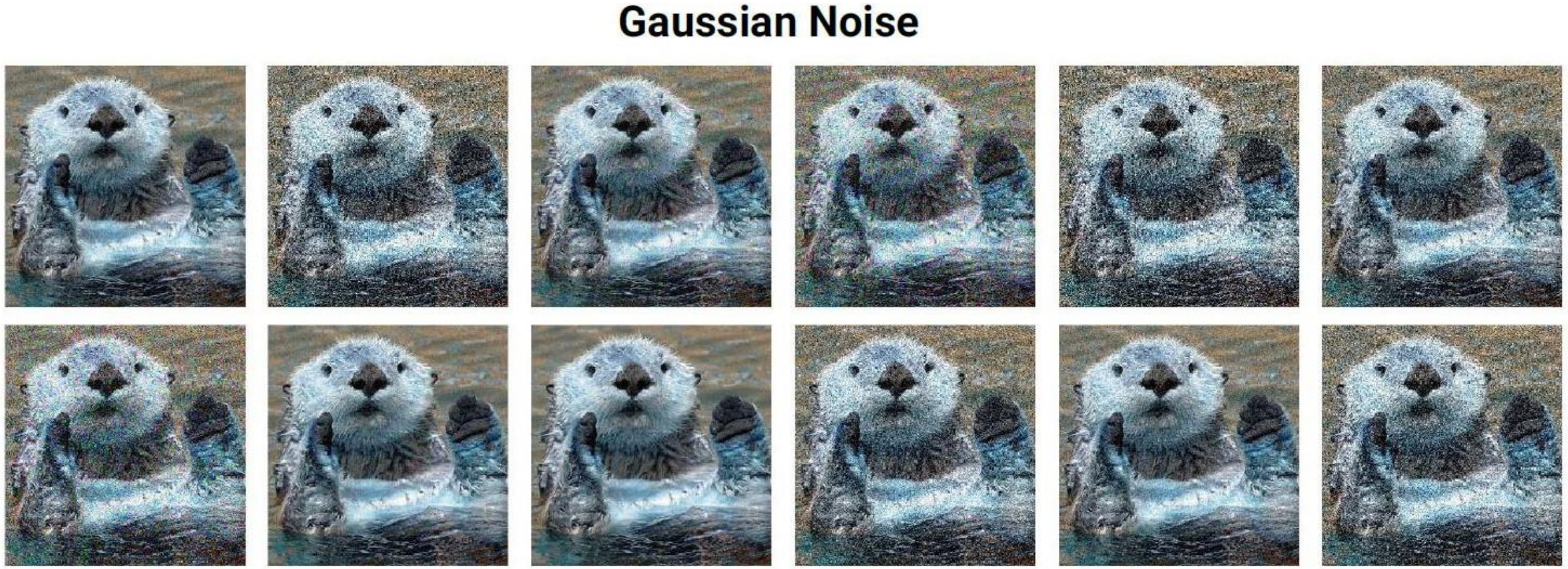

```
iaa.Sharpen(alpha=(0, 0.5), lightness=(0.75, 1.25))
```

Data augmentation
Local Filters




```
iaa.EdgeDetect(alpha=(0, 1.0))
```

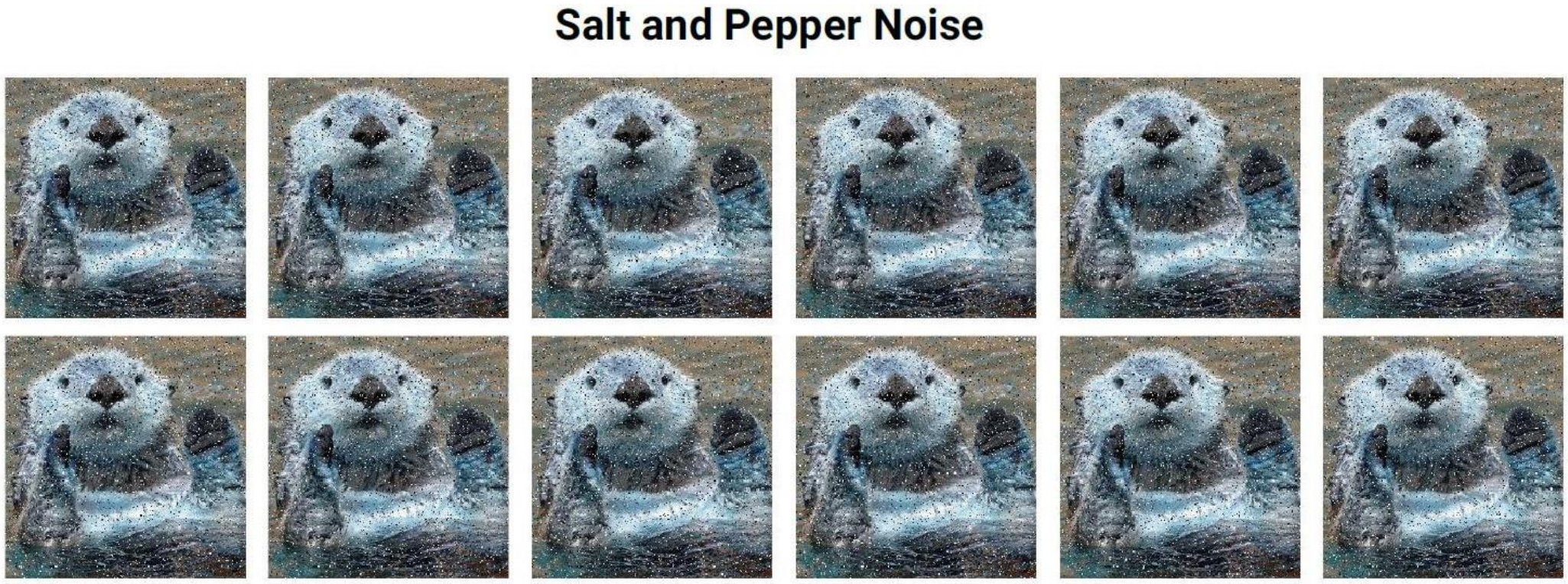
Data augmentation
Adding Noise




```
iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.2*255)
```

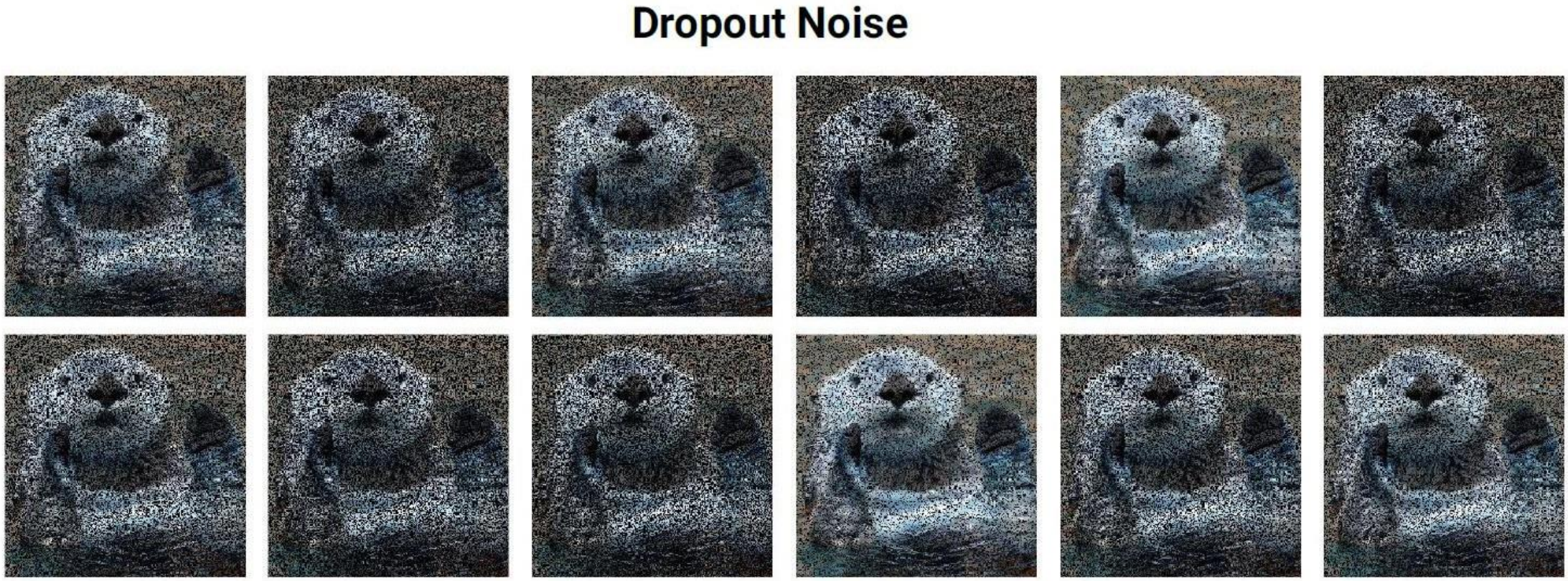
Data augmentation

Adding Noise



`iaa.SaltAndPepper(0.1)`

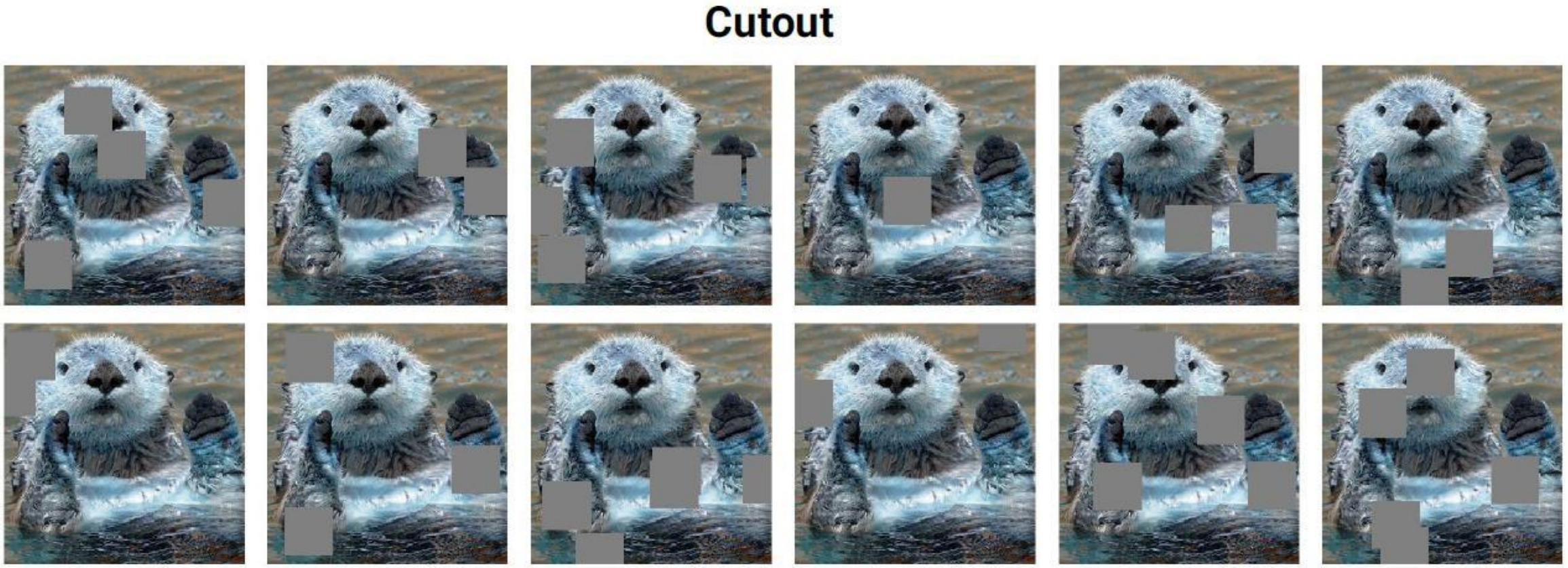
Data augmentation
Adding Noise




```
iaa.Dropout((0.01, 0.5))
```

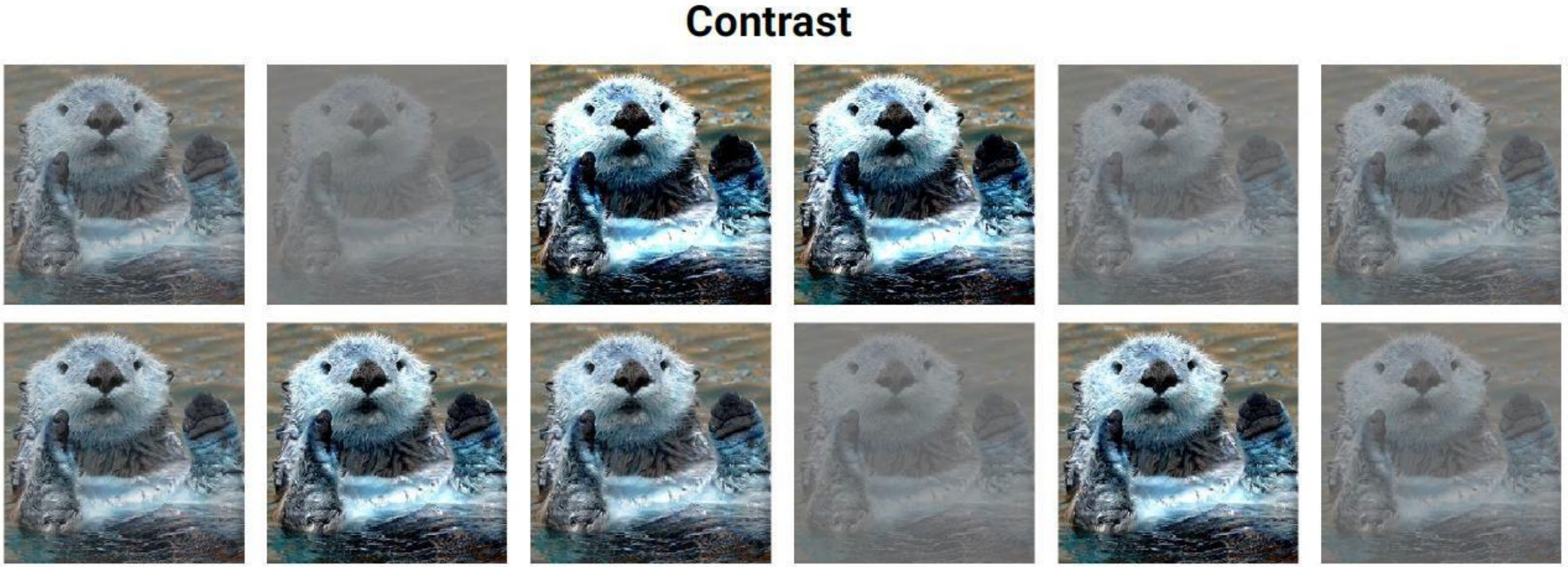
Data augmentation

Adding Noise




```
iaa.Cutout(nb_iterations=(1, 5), size=0.2, squared=False)
```

Data augmentation
Color




```
iaa.LinearContrast((0.1, 2.0), per_channel=0)
```

Data augmentation
Color


```
iaa.Multiply((0.5, 1.5), per_channel=0)
```

Data augmentation
Color

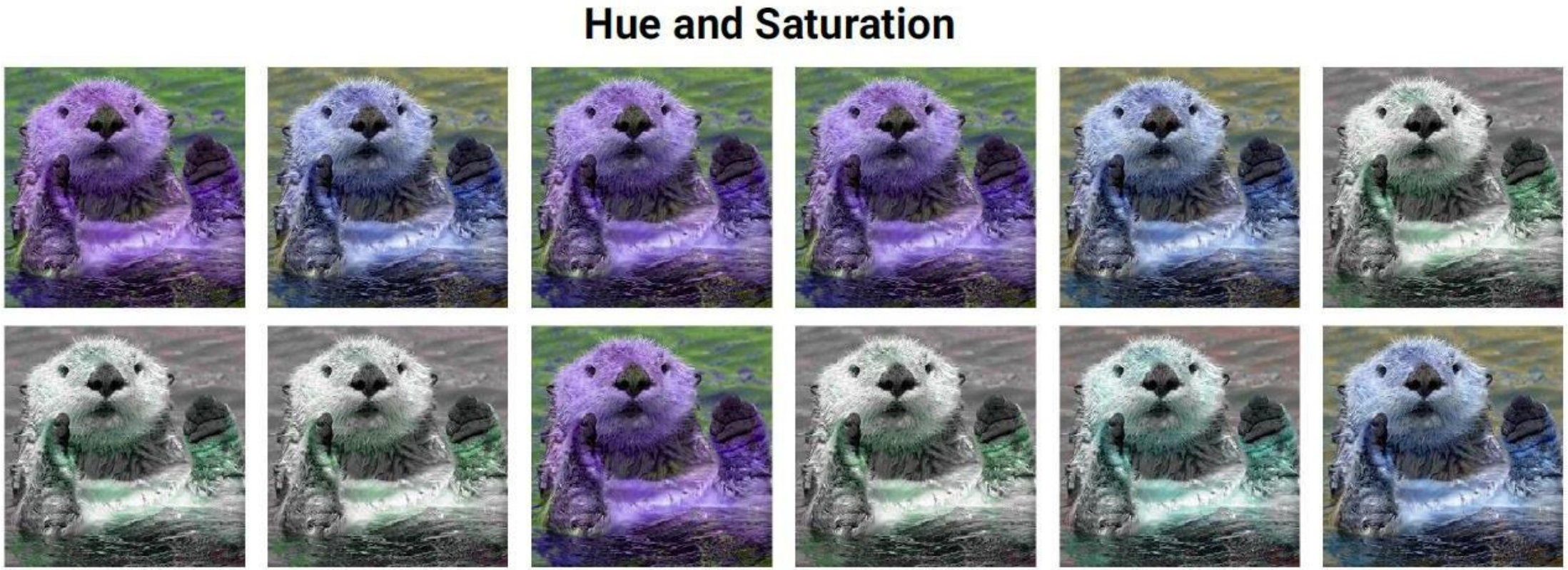

```
iaa.Multiply((0.5, 1.5), per_channel=0.5)
```

Data augmentation

Color

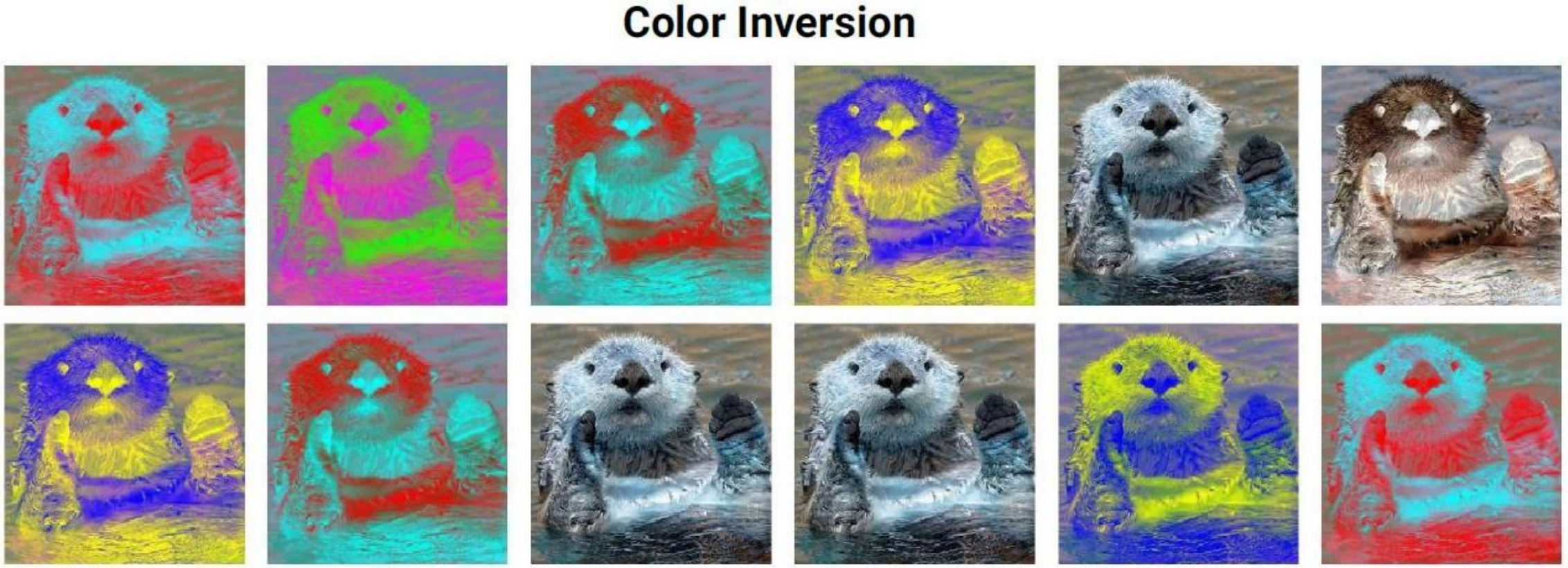

```
iaa.FrequencyNoiseAlpha(exponent=(-4, 0), first=iaa.Multiply((0.5, 1.5), per_channel=True))
```

Data augmentation
Color



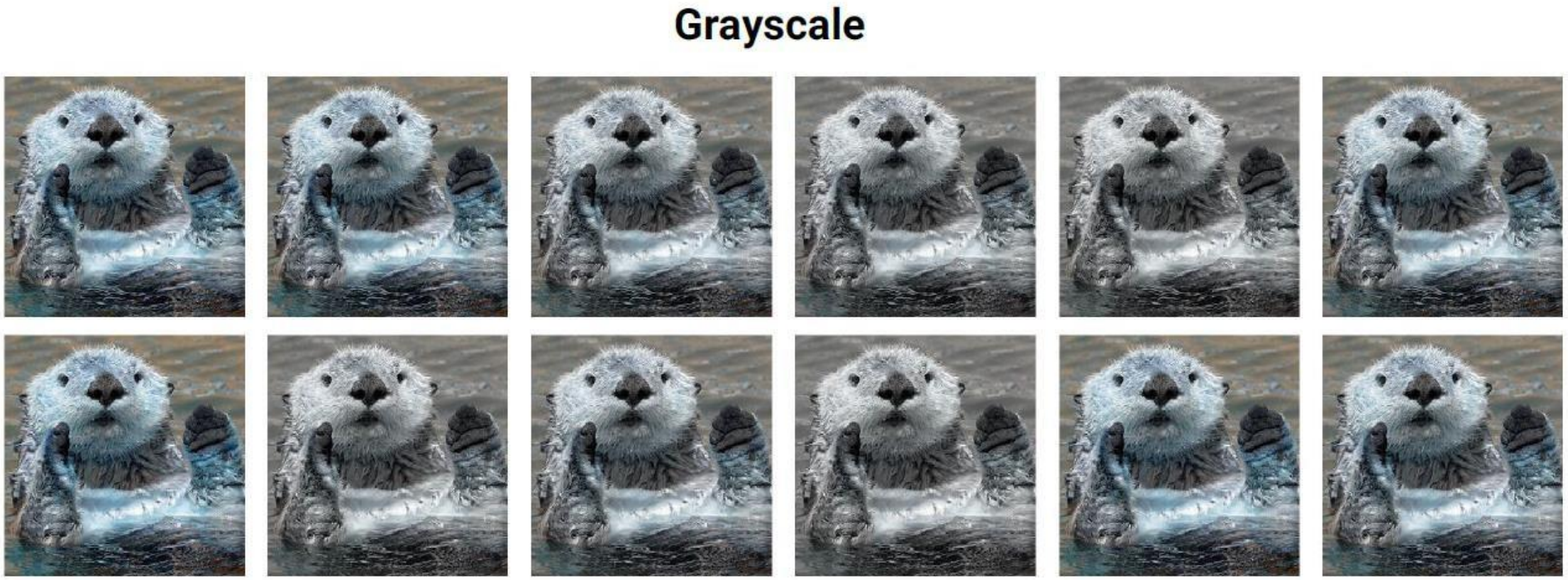

```
iaa.AddToHueAndSaturation((-50, 50))
```

Data augmentation
Color




```
iaa.Invert(0.5, per_channel=0.75)
```

Data augmentation
Color

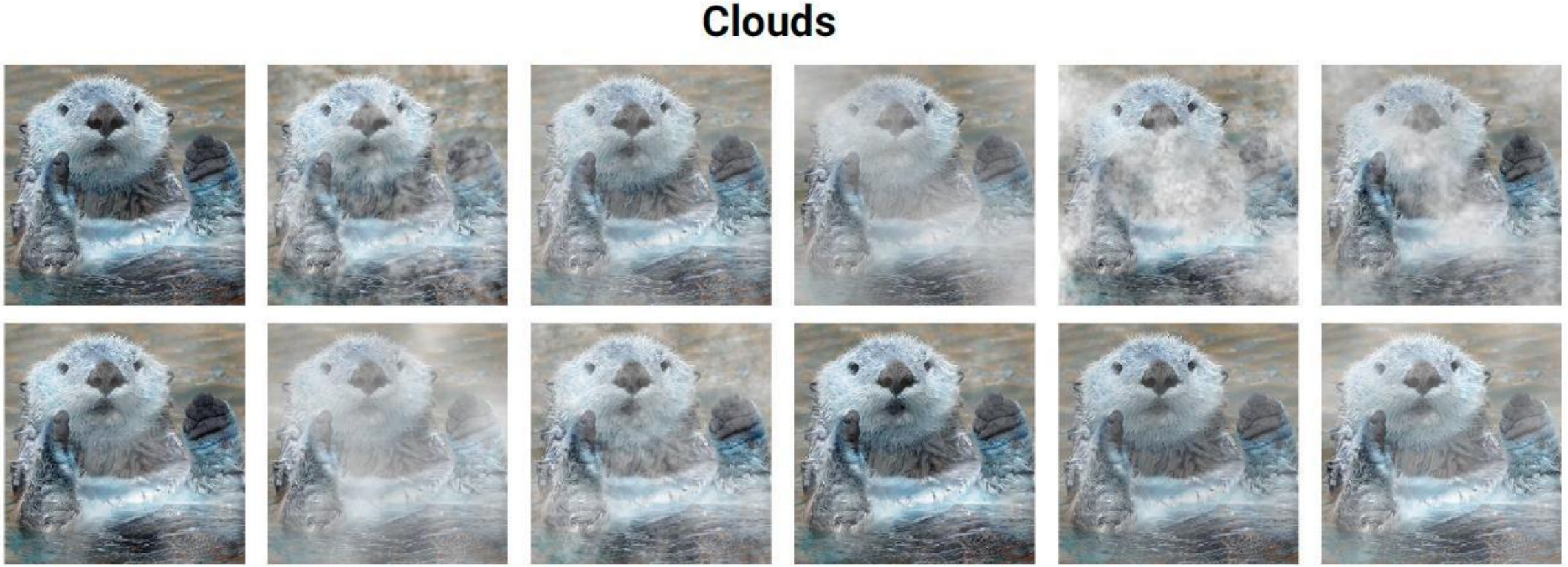



```
iaa.Grayscale(alpha=(0.0, 1.0))
```

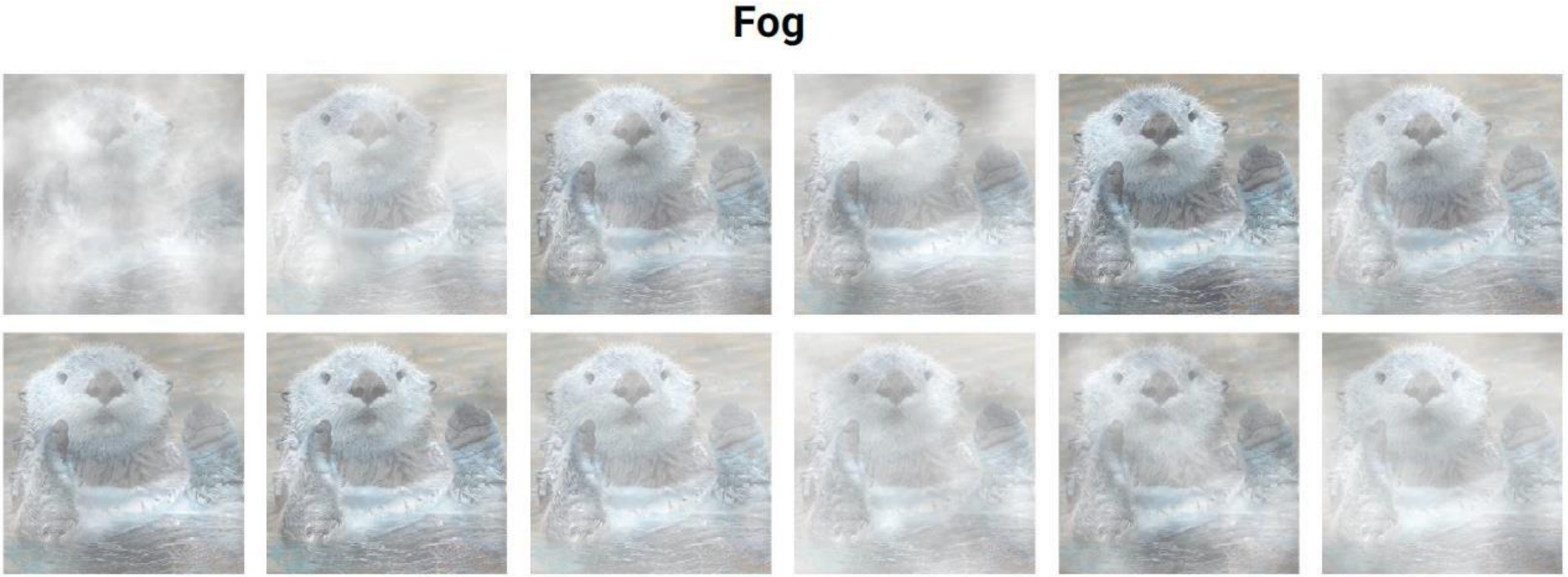
Data augmentation
Weather


```
iaa.FastSnowyLandscape(lightness_threshold=(100, 255),lightness_multiplier=(1.0, 4.0))
```

Data augmentation
Weather

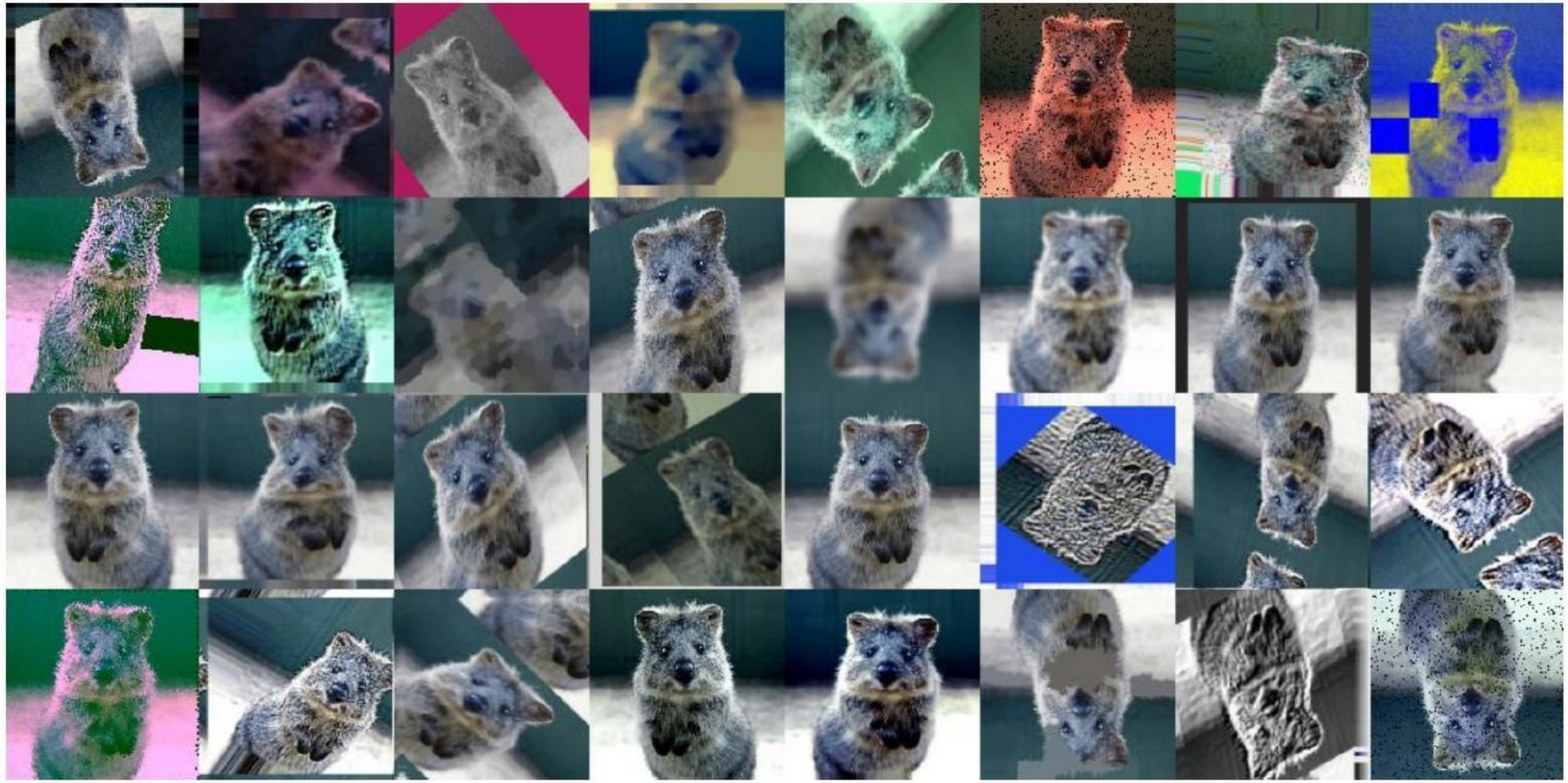


Data augmentation
Weather



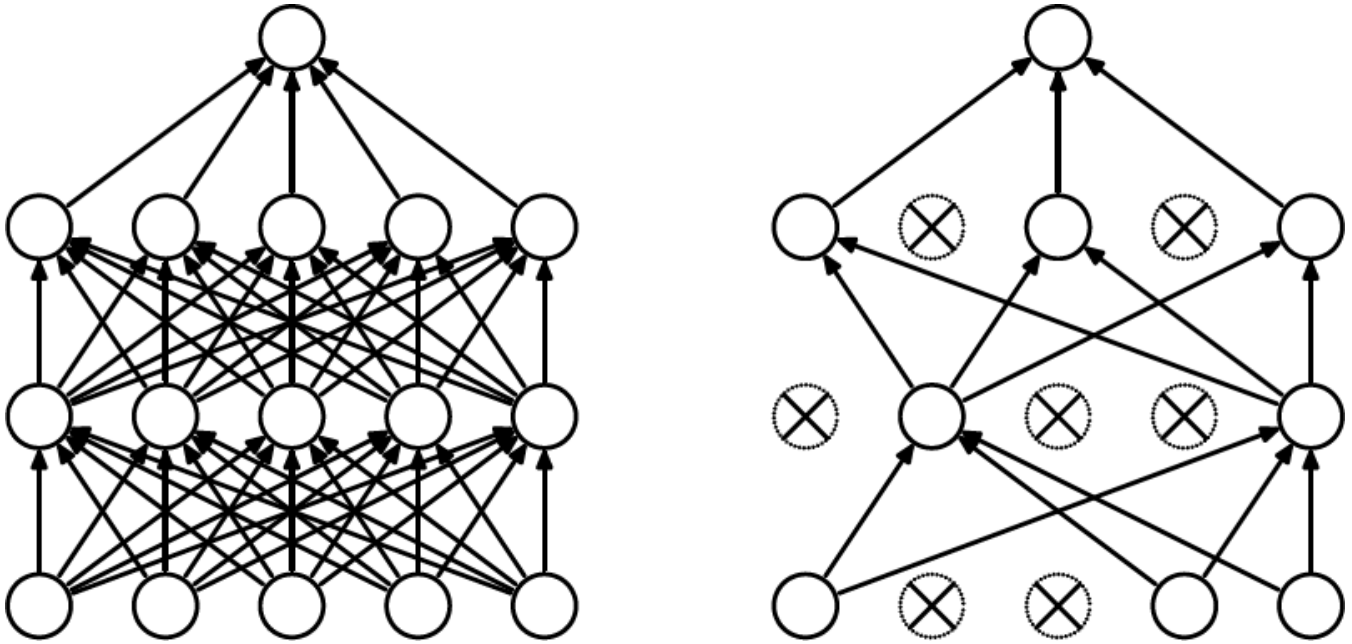
Data augmentation

Random Combinations



Dropout

Noise augmentation



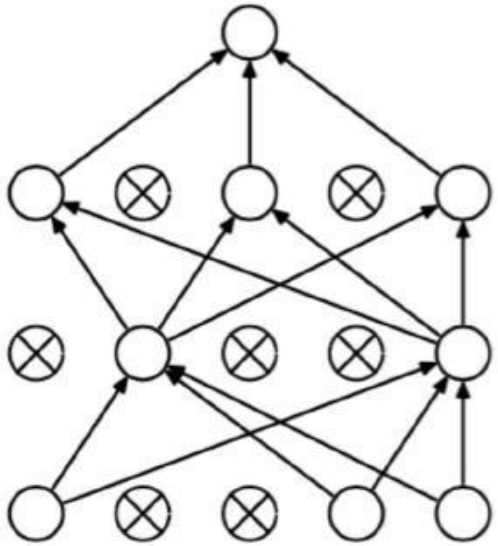
Idea

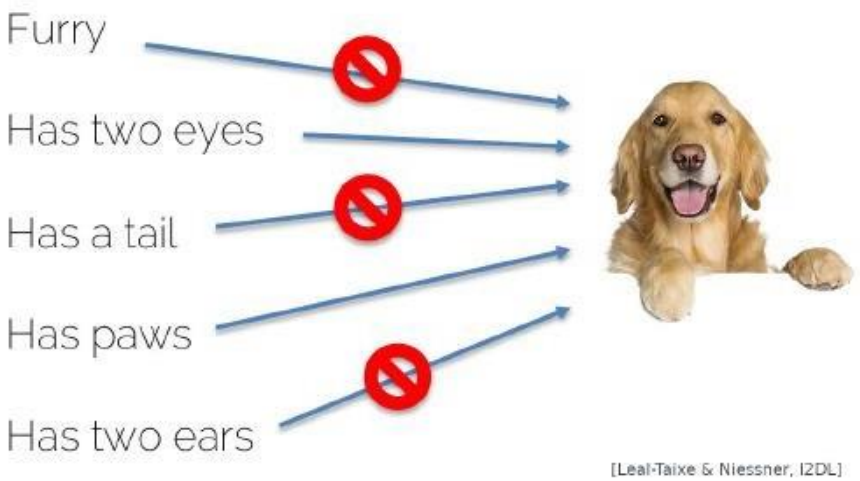
During training, **set neurons to zero** with probability (typically = 0:5)

Each binary mask is one model, changes randomly with every training iteration

Creates **ensemble “on the fly”** from a single network with shared parameters

Dropout
Noise augmentation





Why is this a good idea?

Forces the network to learn a **redundant representation r**egularization

Reduces effective **model capacity** → larger models, longer training

Prevents **co-adaptation** of features (units can't learn to undo output of others)

Requires only **one forward pass at inference** time → Why?

Síivastava, Hinton, Kíizhevsky, Sutskeveí and Salakhutdinov: Díopout: a simple way to píevent neuíal netwoíks fíom oveífitting. JMLR, 2014.

Dropout
Noise augmentation

Let us consider a simple **linear model:**

$$f_{\mathbf{w}}(\mathbf{x}) = w_1 x_1 + w_2 x_2$$

$$f_{\mathbf{w}}(\mathbf{x}, \mathbf{z}) = z_1 w_1 x_1 + z_2 w_2 x_2$$

Assuming $\mu = 0.5$, during tíaining we optimize the **expectation over the ensemble:**

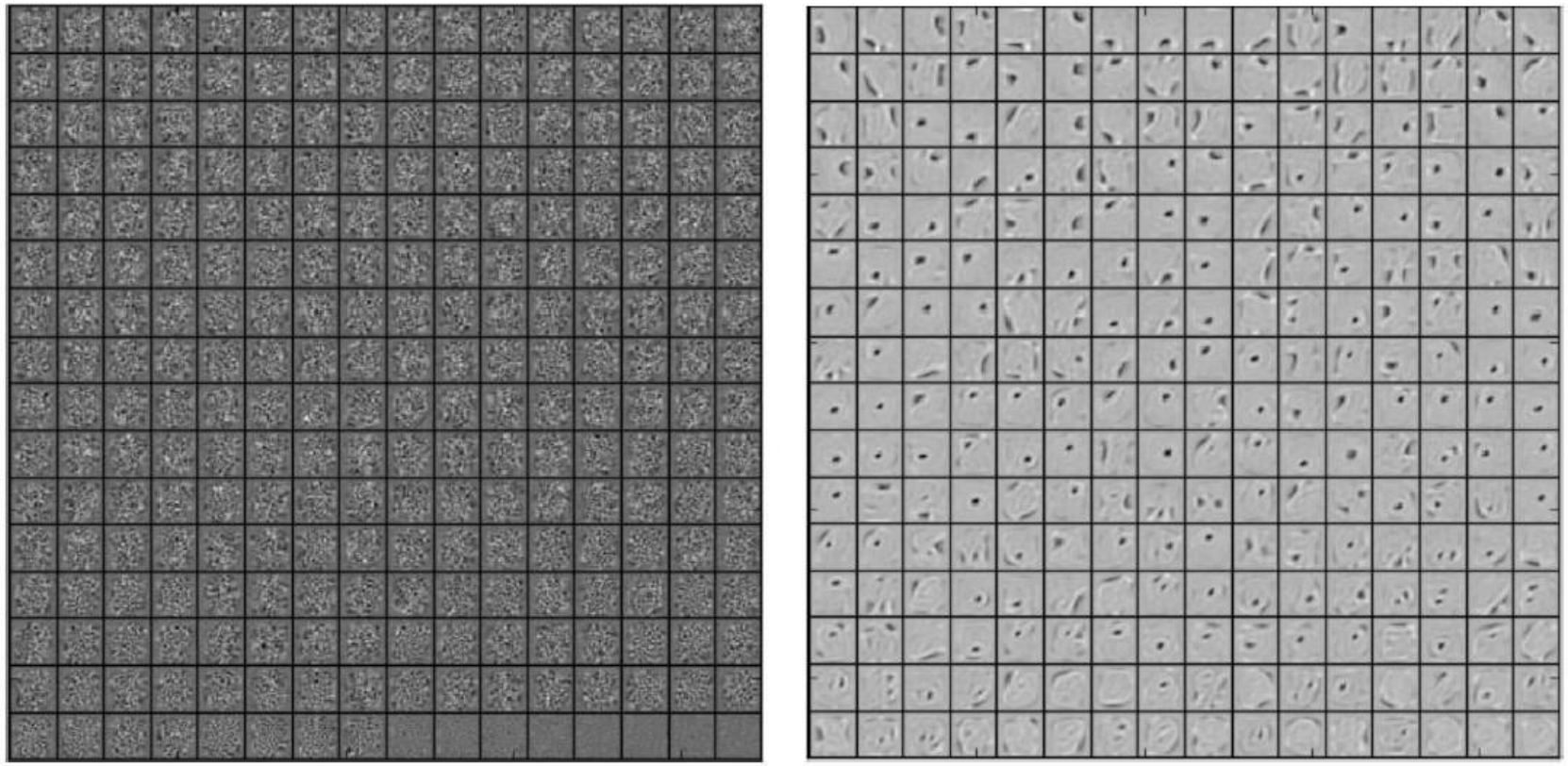
$$\begin{aligned}\mathbb{E}_{\mathbf{z}}[f_{\mathbf{w}}(\mathbf{x}, \mathbf{z})] &= \frac{1}{4}(0+0) + \frac{1}{4}(w_1 x_1 + 0) + \frac{1}{4}(0 + w_2 x_2) + \frac{1}{4}(w_1 x_1 + w_2 x_2) \\ &= \frac{1}{2}(w_1 x_1 + w_2 x_2) = \frac{1}{2} f_{\mathbf{w}}(\mathbf{x})\end{aligned}$$

Thus, at test time, we must **multiply the trained weights** by $1 - \mu$

Remark: This weight scaling inference is only an approximation for non-linear models.

Dropout

Noise augmentation



- Features of an Autoencoder on MNIST with a single hidden layer of 256 ReLUs
- Right: With dropout ⇒ less co-adaptation and thus better generalization

Dropout
Noise augmentation

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

Table 9: Comparison of different regularization methods on MNIST.