Regularization

Rowel Atienza
rowel@eee.upd.edu.ph
University of the Philippines

Regularization

Any modification made to a learning algorithm that reduces the generalization error but not the training error

Objective : Training Error ~ Test Error

A method to prevent the machine learning algorithm from overfitting

Strategy

Add constraints on the parameters
Add extra term or constraints on the loss function
Constraints are sometimes prior knowledge or expressed preference to a specific model
Incorporate inductive bias in the dataset

Key Idea from Maximum A Posteriori

$$\underset{\theta}{\operatorname{argmax}} \log p(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{y}) = \underset{\theta}{\operatorname{argmax}} (\log p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}))$$

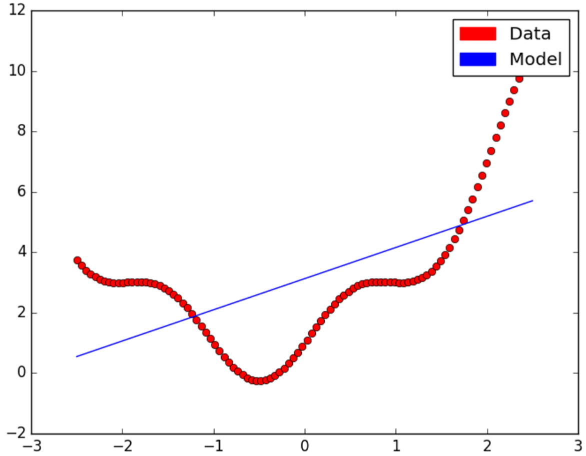
Capacity

Capacity - ability to fit a wide variety of functions

↓ Capacity → Underfitting: ↑ $MSE^{(train)}$, ↑ $MSE^{(test)}$

↑ Capacity → Overfitting: ↓ $MSE^{(train)}$, ↑ $MSE^{(test)}$

✓ Capacity → **Optimal Fit**: ↓ $MSE^{(train)}$, ↓ $MSE^{(test)}$



Underfitting: 1st degree polynomial

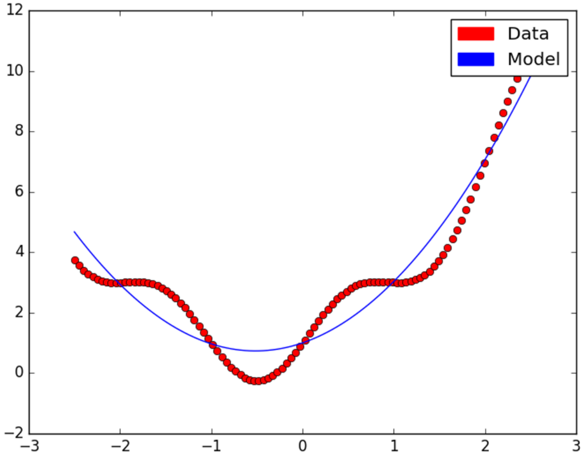
Distribution Function:

Output is second degree polynomial:

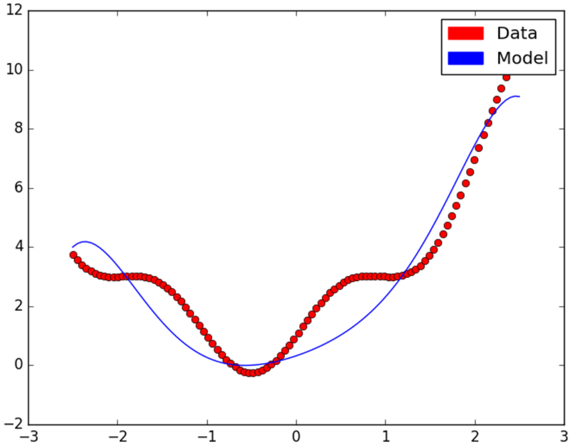
$y = x^2 + x + 1$

Sinusoidal noise is added to output.

Optimal Fit: 2nd degree



Overfitting: 6th degree



Machine Learning - 2 Key Objectives

Make the training error small

Otherwise, the model is underfitting

Make the gap between training and test errors small

Otherwise, the model is overfitting

Loss Penalty Function

Parameter Norm Penalty $\Omega(\boldsymbol{\theta})$ as Regularizer: Limits the capacity of the model

$$L'(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}) = L(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

where hyperparameter $\alpha \in [0,\infty)$ weights the contribution of the $\Omega(\boldsymbol{\theta})$ to the objective function L

Regularization on weights only not biases; Regularizing biases introduces underfitting: $L'(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}) = L(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y}) + \alpha\Omega(\boldsymbol{w})$

L2 Regularization

Weight decay: $\Omega(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$

$$L'(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) = L(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) + \alpha\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

Assuming: $\mathcal{N}(\boldsymbol{w}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}e^{-\frac{1}{2}\left((\boldsymbol{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\right)}$

Modified gradient update:

$$\boldsymbol{\nabla}'_{\boldsymbol{w}}L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\nabla}_{\boldsymbol{w}}L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) + \alpha\boldsymbol{w}$$

$$\boldsymbol{w} = \boldsymbol{w} - \in \boldsymbol{\nabla}'_{\boldsymbol{w}}L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y})$$

Also known as: Ridge Regression or Tikhonov Regularization

L2 Regularization on Linear Regression

MSE: $L = (\boldsymbol{xw} - \boldsymbol{y})^T(\boldsymbol{xw} - \boldsymbol{y})$

Regularized MSE:

$$L'(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{xw} - \boldsymbol{y})^T(\boldsymbol{xw} - \boldsymbol{y}) + \alpha\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$
$$\boldsymbol{\nabla}'_{\boldsymbol{w}}\boldsymbol{L}(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = (2\boldsymbol{x}^T\boldsymbol{x} + \alpha\boldsymbol{I})\boldsymbol{w} - 2\boldsymbol{x}^T\boldsymbol{y} = 0$$
$$\boldsymbol{w} = (\boldsymbol{x}^T\boldsymbol{x} + \alpha\boldsymbol{I})^{-1}\boldsymbol{x}^T\boldsymbol{y}$$

$\alpha\boldsymbol{I}$ makes the learning algorithm think that $\boldsymbol{x}$ has high variance thus shrinking the weights; weights closer to zero but not zero.

L1 Regularization

Weight decay: $\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 = \sum_i |\boldsymbol{w}_i|$

$$L'(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) + \alpha\|\boldsymbol{w}\|_1$$

$$\boldsymbol{\nabla}'_{\boldsymbol{w}} L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\nabla}_{\boldsymbol{w}} L(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) + \alpha sign(\boldsymbol{w})$$

where $sign(\boldsymbol{w})$ is the sign of $\boldsymbol{w}$ element-wise

tries to zero weights creating a sparse feature space (smaller model): A form of feature selection

Dataset Augmentation – Inductive Bias on Dataset

Increasing the amount of training data by creating fake data

Effective in computer vision (eg image of a dog rotated, resized, translated, etc) and speech

Original

Flipped

Rotated

Dataset Augmentation

Some image data are not amenable to geometric transformation as the meaning is altered

For example,

b	q	d
Orig	Rot/Flipped	Mirrored

Data Augmentation

Other operations

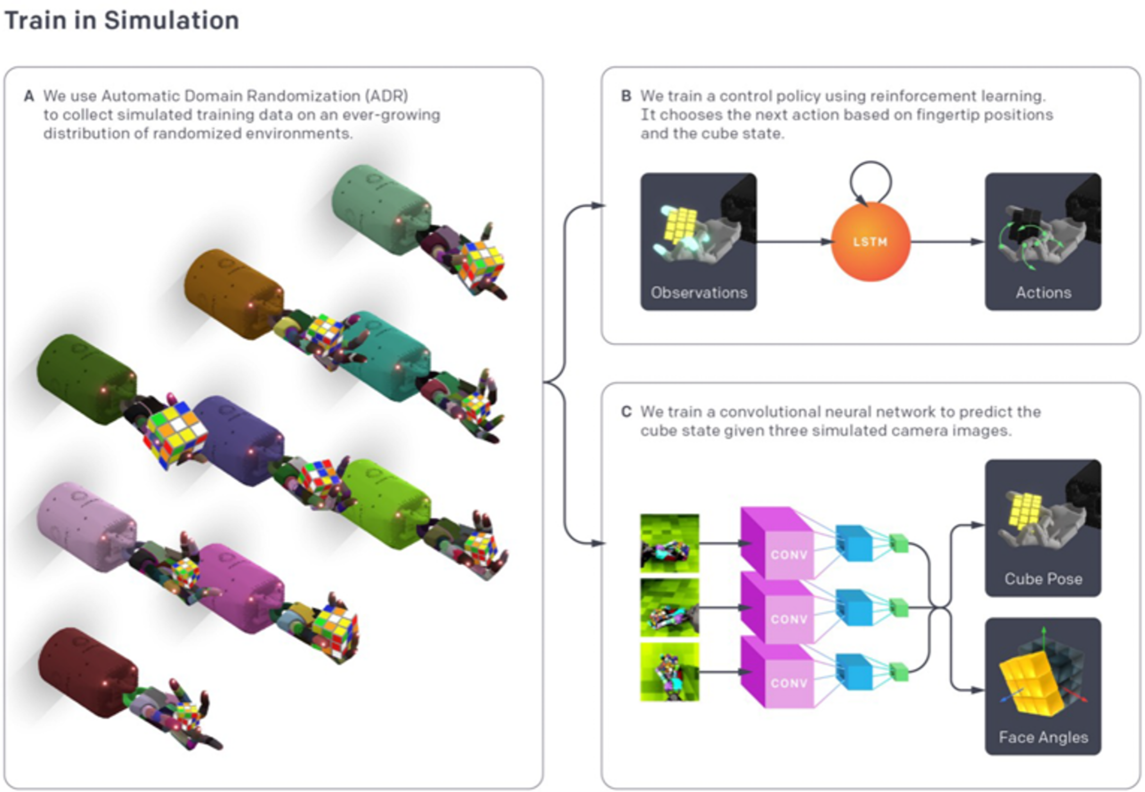
1. Scaling
2. Translating
3. Minor distortion
4. Normalization
5. ZCA Whitening
6. Color Jitter
7. Shearing
8. Polarize

Data Augmentation

Other methods of Dataset Augmentation: Adding noise to input

Neural Networks are inherently not insensitive to noise
Training a NN with a small amount of noise added to the data can make the network more robust

Automatic Domain Randomization (ADR)



Noise as Weights Regularizer

Noise can also be added to weights

In Bayesian Approach, model weights are stochastic

Adding noise is one way to model stochasticity

For example, MSE: $L = (\boldsymbol{xw} - \boldsymbol{y})^T(\boldsymbol{xw} - \boldsymbol{y})$

Suppose the weights are corrupted:

$$\boldsymbol{w} = (\boldsymbol{x}^T\boldsymbol{x} + \alpha\boldsymbol{I})^{-1}\boldsymbol{x}^T\boldsymbol{y} + noise$$

The constraints drive the values of $\boldsymbol{w}$ to a flat region such that a small perturbation will not alter the prediction

Noise on Output

Targets might be incorrectly labeled

Training might not converge

On logistic classification with k output values, one way to handle noise is by modifying probabilities so that we have soft targets

0 as $\frac{\epsilon}{k-1}$

1 as $1 - \epsilon$

Soft targets can converge with softmax output and cross entropy loss

Multi-task Learning

Same data is used to learn multiple tasks

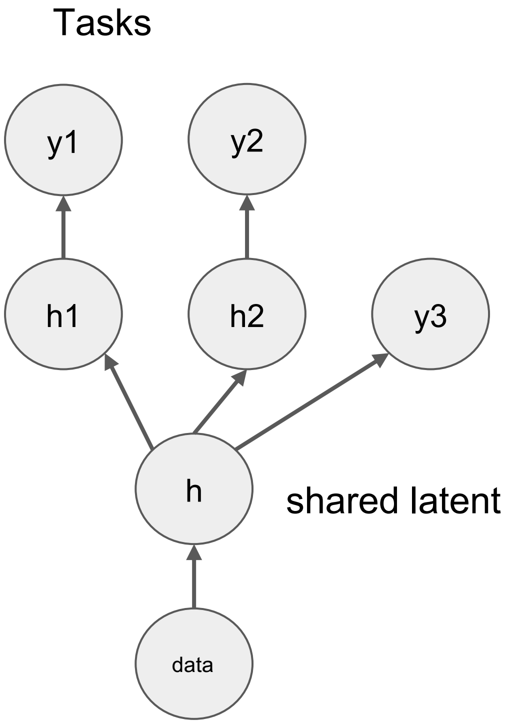
This is a form of regularization since the network is subjected to soft constraints

Shared parameters across multiple tasks

In the factors that explain variations of observed data across multiple tasks, some are shared among multiple tasks

$y1$ may be classification, $y2$ may be bounding box detection, $y3$ may segmentation

Example: Mask RCNN



Early Stopping

Initially, validation error decreases as training error decreases

As training continues, the training error decreases while the validation error increases; network is overfitting

Early Stopping stops the training and saves the parameters when the validation error starts to increase

Early Stopping is a form of regularization since it limits the parameter space

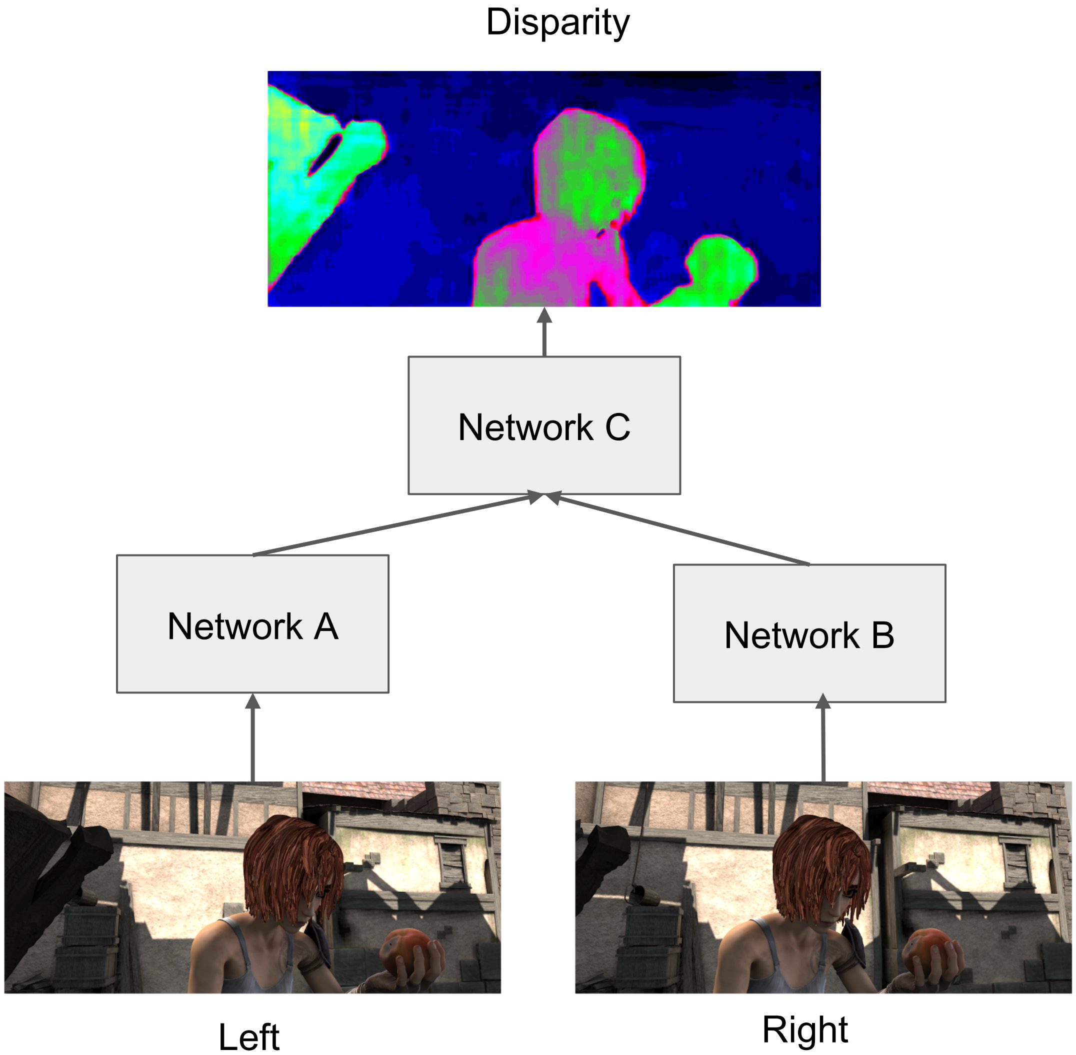
Parameter Sharing

Two different networks doing similar tasks can share parameters

Very common in Siamese Networks

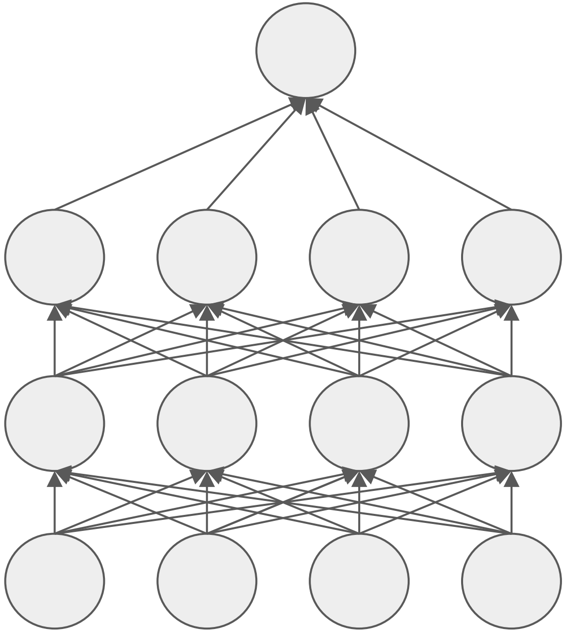
Example, Networks A and B share the same set of parameters in a Siamese Network to perform disparity estimation

Forces Networks A and B to be more robust by forcing them to work on different images from the same distribution

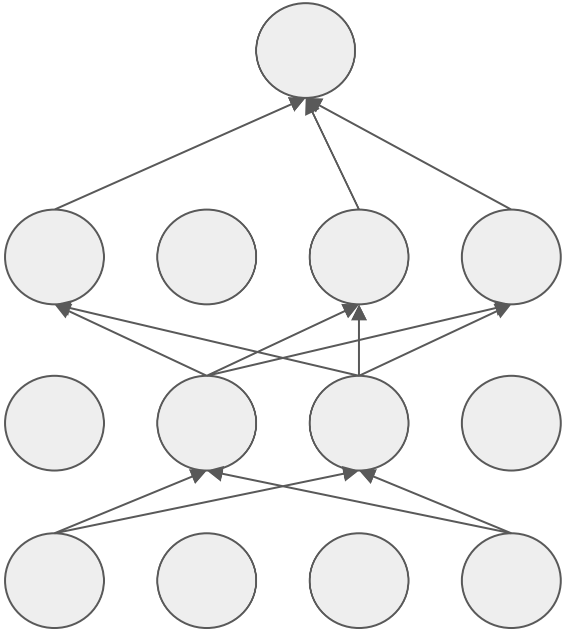


Dropout

Strongly inspired by biological processes

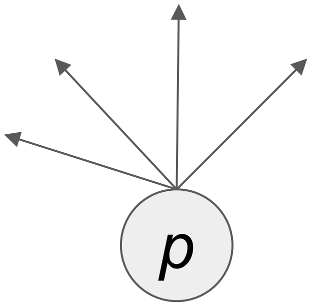


Standard Deep Neural Network

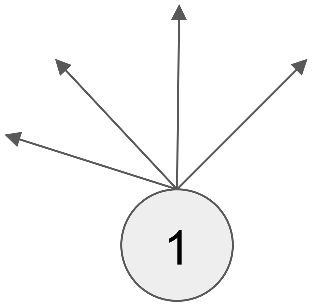


Standard Deep Neural Network
With Dropout

Dropout



During training, a neuron will be included in the network with probability p



During test, a neuron will always be included in the network

Dropout

Can be interpreted as a way of regularizing a neural network by adding noise to its hidden units: aim is to minimize loss function stochastically under a noise distribution

Model: Neural Network with L hidden layers

$l \in [1,L]$: the index of the hidden layer

$\mathbf{z}^{(l)}$: vector input to layer l

$\mathbf{a}^{(l)}$: vector output of layer l ($\mathbf{a}^{(0)} = \mathbf{x}$ is the input)

$\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$: the weights and biases at layer l

Dropout

Hidden layer unit:

$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)}\mathbf{a}^{(l)} + b_i^{(l+1)}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

where f() is the activation function (eg relu, sigmoid, softmax)

Dropout

With dropout, the new model

$$r_j^{(l)} = \text{Bernoulli}(p)$$

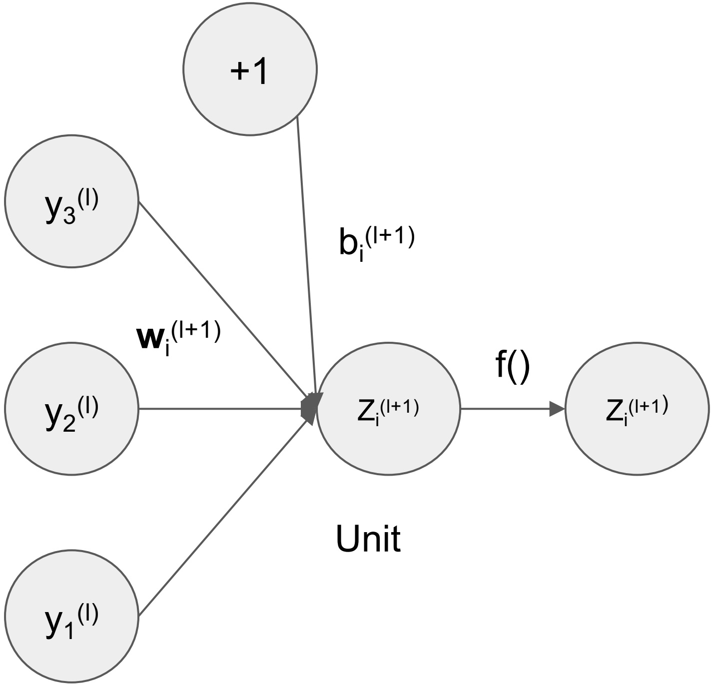
$$\mathbf{a}'^{(l)} = \mathbf{r}^{(l)} \circledcirc \mathbf{a}^{(l)}$$

$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \mathbf{a}'^{(l)} + b_i^{(l+1)}$$

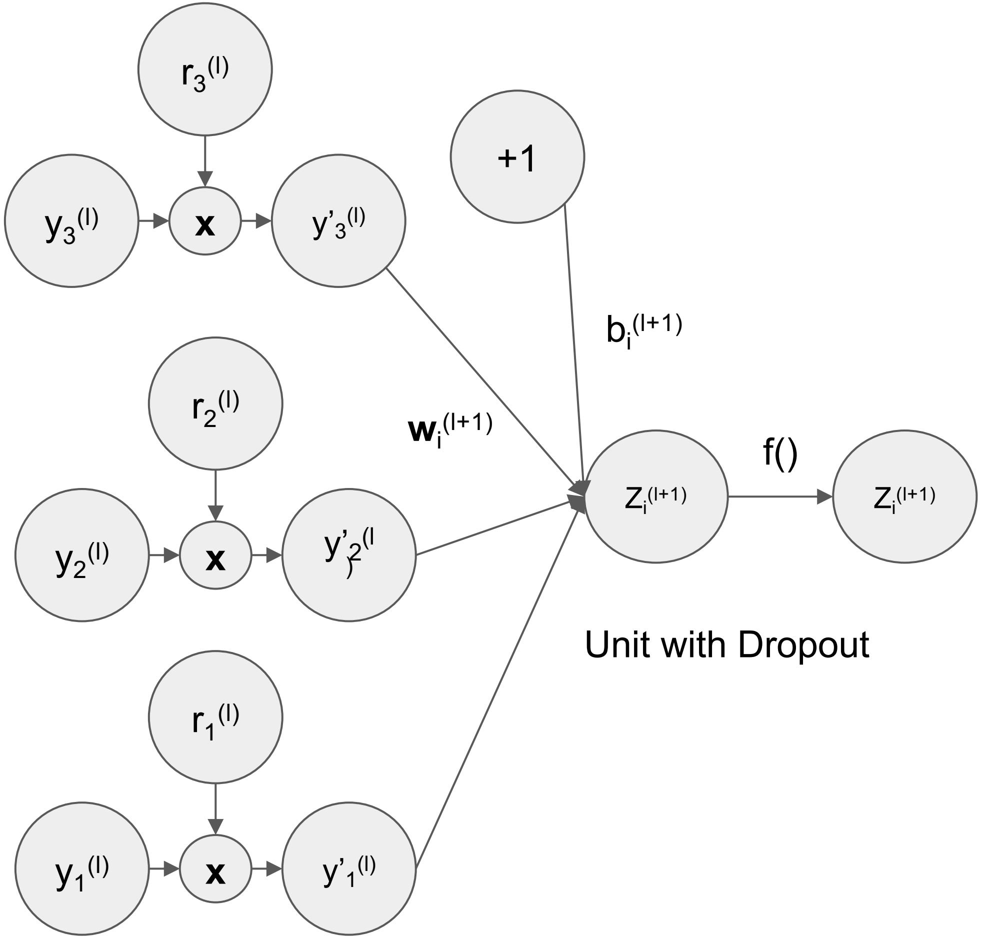
$$y_i^{(l+1)} = f(z_i^{(l+1)})$$

During test, the dropout is removed from the neural network (all units are used)

Dropout

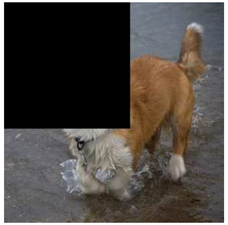


Unit



Unit with Dropout

Regional Dropout

	ResNet-50	Mixup [47]	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)**

MixUp

Zhang, Hongyi, et al. "mixup: Beyond empirical risk minimization." *arXiv preprint arXiv:1710.09412* (2017).

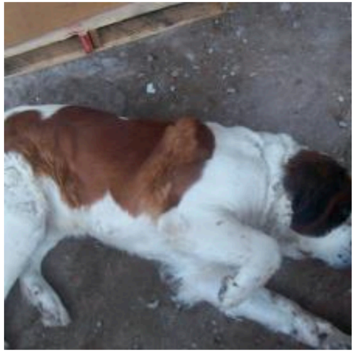
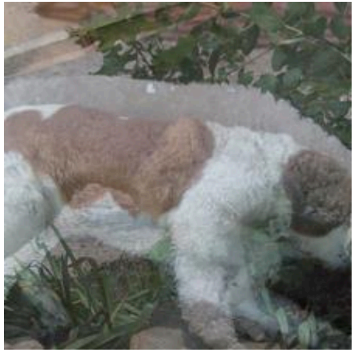
Given 2 data points (x_i, y_i) and (x_j, y_j), a new target is synthesized:

$$x = \lambda x_i + (1 - \lambda) x_j$$

$$y = \lambda y_i + (1 - \lambda) y_j$$

Where $\lambda \sim Beta(\alpha, \alpha)$ and $\alpha = [0, \infty)$

$$\lambda = [0,1]$$

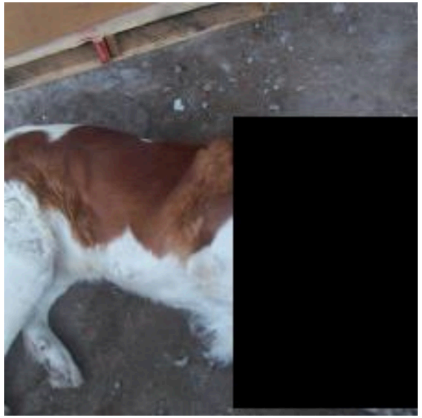
λ  $+(1 - \lambda)$  $=$ 

CutOut

DeVries, Terrance, and Graham W. Taylor. "Improved regularization of convolutional neural networks with cutout." *arXiv preprint arXiv:1708.04552* (2017).

A square region with coordinates $\left((x_{ul}, y_{ul}), (x_{ul} + d, y_{ul} + d)\right)$ with dimensions (d, d) is cropped out

The coordinates are randomly sampled: $x_{ul} \sim [-d, +d]$ and $y_{ul} \sim [-d, +d]$



CutMix

Given 2 data points (x_i, y_i) and $\left(x_j, y_j\right)$, a new target is synthesized:

$$x = \boldsymbol{M} \odot x_i + (\mathbf{1} - \boldsymbol{M}) x_j$$

$$y = \lambda y_i + (1 - \lambda) y_j$$

Where $\lambda \sim Beta(\alpha, \alpha)$ and $\alpha = [0, \infty), \lambda = [0,1], x_i, x_j \in \mathbb{R}^{W \times H \times C}, \boldsymbol{M} \in \mathbb{R}^{W \times H}$

Assuming $\boldsymbol{M}$ is initially all 1's (ie $\mathbf{1}$)

A bounding box $\boldsymbol{B} = (x, y, w, h)$ choose the region of 0's in $\boldsymbol{M}$ such that:

$$x \sim [0, W]$$

$$y \sim [0, H]$$

$$w = W\sqrt{1 - \lambda}$$

$$h = H\sqrt{1 - \lambda}$$

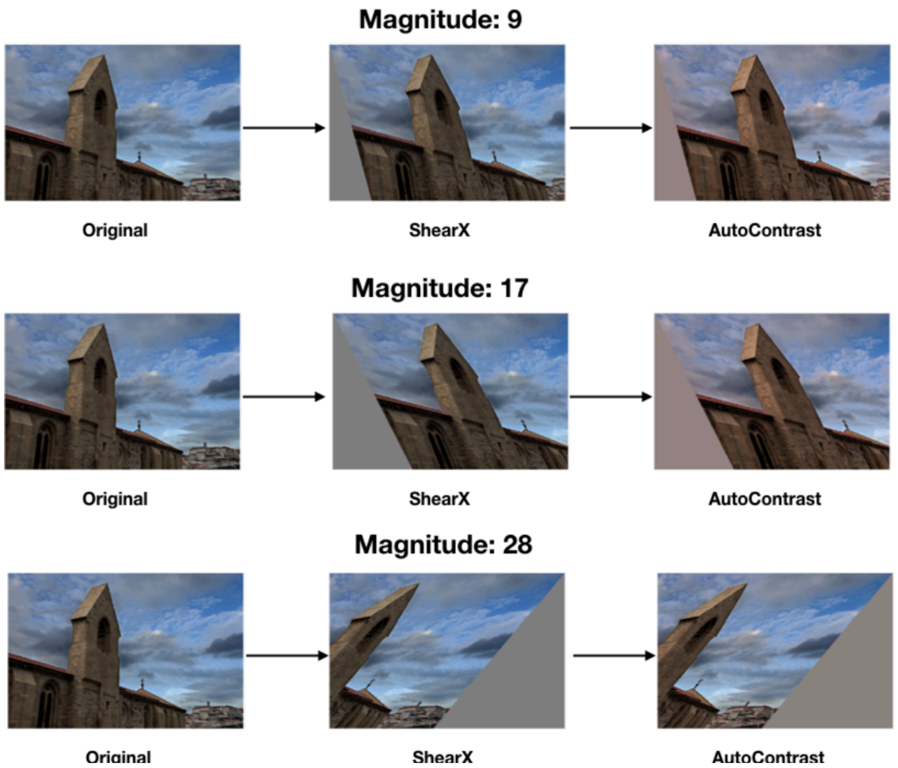
CutMix

Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

$\boldsymbol{M}$ 

$+(1-\boldsymbol{M})$  $=$ 

Policy-Based Auto Augmentation



AutoAugment

Cubuk, Ekin D., et al. "Autoaugment: Learning augmentation strategies from data." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019.

Cubuk, Ekin D., et al. "Randaugment: Practical automated data augmentation with a reduced search space." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020.

	Operation 1	Operation 2
Sub-policy 0	(Posterize,0.4,8)	(Rotate,0.6,9)
Sub-policy 1	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 2	(Equalize,0.8,8)	(Equalize,0.6,3)
Sub-policy 3	(Posterize,0.6,7)	(Posterize,0.6,6)
Sub-policy 4	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 5	(Equalize,0.4,4)	(Rotate,0.8,8)
Sub-policy 6	(Solarize,0.6,3)	(Equalize,0.6,7)
Sub-policy 7	(Posterize,0.8,5)	(Equalize,1.0,2)
Sub-policy 8	(Rotate,0.2,3)	(Solarize,0.6,8)
Sub-policy 9	(Equalize,0.6,8)	(Posterize,0.4,6)
Sub-policy 10	(Rotate,0.8,8)	(Color,0.4,0)
Sub-policy 11	(Rotate,0.4,9)	(Equalize,0.6,2)
Sub-policy 12	(Equalize,0.0,7)	(Equalize,0.8,8)
Sub-policy 13	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 14	(Color,0.6,4)	(Contrast,1.0,8)
Sub-policy 15	(Rotate,0.8,8)	(Color,1.0,2)
Sub-policy 16	(Color,0.8,8)	(Solarize,0.8,7)
Sub-policy 17	(Sharpness,0.4,7)	(Invert,0.6,8)
Sub-policy 18	(ShearX,0.6,5)	(Equalize,1.0,9)
Sub-policy 19	(Color,0.4,0)	(Equalize,0.6,3)
Sub-policy 20	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 21	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 22	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 23	(Color,0.6,4)	(Contrast,1.0,8)
Sub-policy 24	(Equalize,0.8,8)	(Equalize,0.6,3)

Table 9. AutoAugment policy found on reduced ImageNet.

Reference

Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, 2016, http://www.deeplearningbook.org

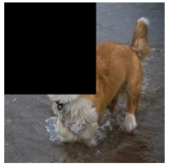
Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Srivastava, et. al. Journal of Machine Learning, 2014

In Summary

More than the architecture, there is so much improvement that can be achieved from using a proper regularizer

More regularization methods to be invented down the road

No single regularizer is superior over the rest

	ResNet-50	Mixup [47]	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)**

Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.