

Lecture 7: Training Neural Networks

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 1

April 25, 2023

Activation Functions

Fei-Fei Li, Yunzhu Li, Ruohan Gao

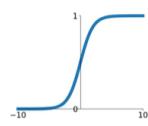
Lecture 7 - 12

April 25, 2023

Activation Functions

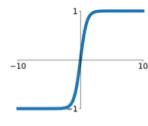
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



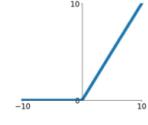
tanh

$$\tanh(x)$$



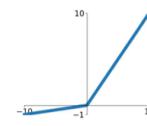
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

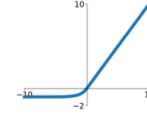


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Fei-Fei Li, Yunzhu Li, Ruohan Gao

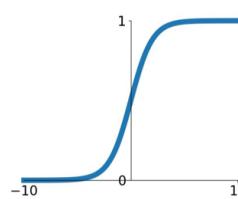
Lecture 7 - 14

April 25, 2023

Activation Functions

$$\sigma(x) = 1/(1 + e^{-x})$$

- Squashes numbers to range [0,1]
- Historically popular since they have nice interpretation as a saturating "firing rate" of a neuron



Sigmoid

3 problems:

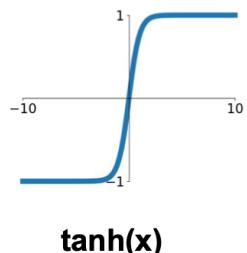
1. Saturated neurons "kill" the gradients
2. Sigmoid outputs are not zero-centered
3. `exp()` is a bit compute expensive

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 32

April 25, 2023

Activation Functions



- Squashes numbers to range [-1,1]
- zero centered (nice)
- still kills gradients when saturated :)

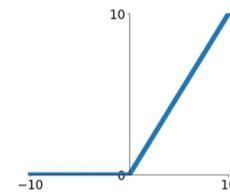
[LeCun et al., 1991]

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 33

April 25, 2023

Activation Functions



ReLU
(Rectified Linear Unit)

- Computes $f(x) = \max(0,x)$
- Does not saturate (in +region)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice (e.g. 6x)

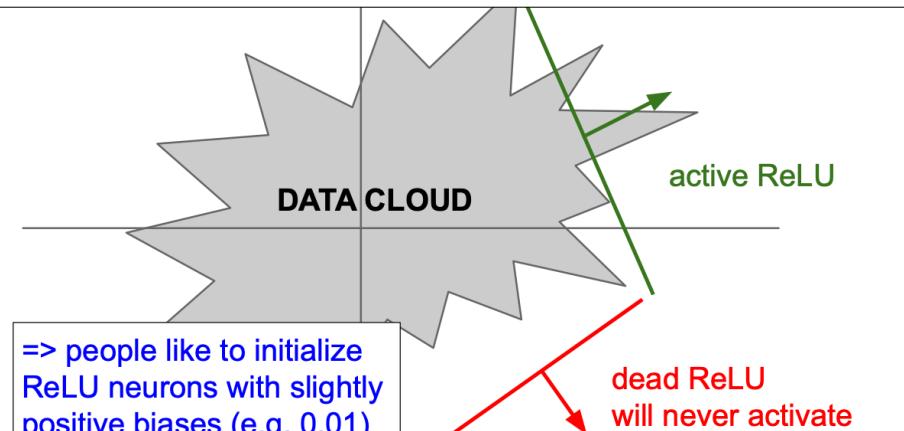
- Not zero-centered output
- An annoyance:

hint: what is the gradient when $x < 0$?

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 36

April 25, 2023

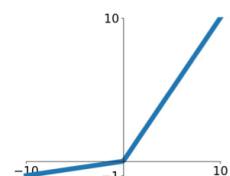


Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 39

April 25, 2023

Activation Functions



Leaky ReLU
 $f(x) = \max(0.01x, x)$

[Maas et al., 2013]
[He et al., 2015]

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- will not “die”.

Parametric Rectifier (PReLU)
 $f(x) = \max(\alpha x, x)$

backprop into α (parameter)

Fei-Fei Li, Yunzhu Li, Ruohan Gao

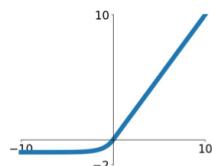
Lecture 7 - 41

April 25, 2023

Activation Functions

[Clevert et al., 2015]

Exponential Linear Units (ELU)



- All benefits of ReLU
- Closer to zero mean outputs
- Negative saturation regime compared with Leaky ReLU adds some robustness to noise

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

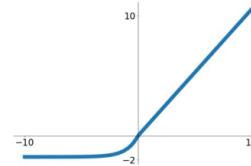
(Alpha default = 1)

- Computation requires `exp()`

Activation Functions

[Klambauer et al. ICLR 2017]

Scaled Exponential Linear Units (SELU)



$$f(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^x - 1) & \text{otherwise} \end{cases}$$

$\alpha = 1.6732632423543772848170429916717$
 $\lambda = 1.0507009873554804934193349852946$

Maxout “Neuron”

[Goodfellow et al., 2013]

- Does not have the basic form of dot product -> nonlinearity
- Generalizes ReLU and Leaky ReLU
- Linear Regime! Does not saturate! Does not die!

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

Problem: doubles the number of parameters/neuron :(

TLDR: In practice:

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU / SELU
 - To squeeze out some marginal gains
- Don't use sigmoid or tanh

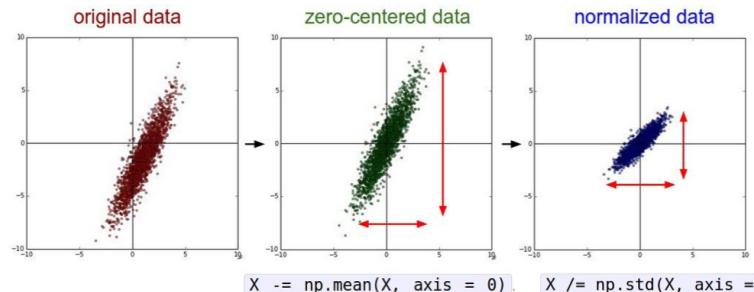
Data Preprocessing

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 46

April 25, 2023

Data Preprocessing



(Assume X [NxD] is data matrix, each example in a row)

Fei-Fei Li, Yunzhu Li, Ruohan Gao

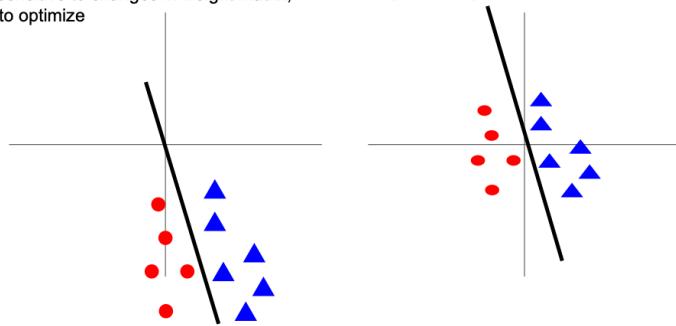
Lecture 7 - 49

April 25, 2023

Data Preprocessing

Before normalization: classification loss very sensitive to changes in weight matrix; hard to optimize

After normalization: less sensitive to small changes in weights; easier to optimize



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 51 April 25, 2023

TLDR: In practice for Images: center only

e.g. consider CIFAR-10 example with [32,32,3] images

- Subtract the mean image (e.g. AlexNet)
(mean image = [32,32,3] array)
- Subtract per-channel mean (e.g. VGGNet)
(mean along each channel = 3 numbers)
- Subtract per-channel mean and
Divide by per-channel std (e.g. ResNet)
(mean along each channel = 3 numbers)

Not common
to do PCA or
whitening

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 52

April 25, 2023

Weight Initialization

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 53

April 25, 2023

- First idea: **Small random numbers**
(gaussian with zero mean and 1e-2 standard deviation)

```
W = 0.01 * np.random.randn(Din, Dout)
```

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 55

April 25, 2023

- First idea: **Small random numbers**
(gaussian with zero mean and 1e-2 standard deviation)

```
W = 0.01 * np.random.randn(Din, Dout)
```

Works ~okay for small networks, but problems with deeper networks.

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 56

April 25, 2023

Weight Initialization: Activation statistics

```
dims = [4096] * 7      Forward pass for a 6-layer
hs = []                  net with hidden size 4096
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

What will happen to the activations for the last layer?

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 57

April 25, 2023

Weight Initialization: Activation statistics

```

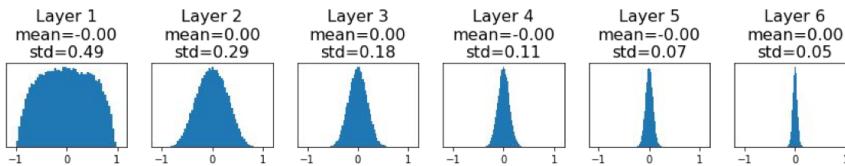
dims = [4096] * 7    Forward pass for a 6-layer
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)

```

All activations tend to zero for deeper network layers

Q: What do the gradients dL/dW look like?

A: All zero, no learning = (



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 59

April 25, 2023

Weight Initialization: Activation statistics

```

dims = [4096] * 7    Increase std of initial
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.05 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)

```

What will happen to the activations for the last layer?

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 60

April 25, 2023

Weight Initialization: Activation statistics

```

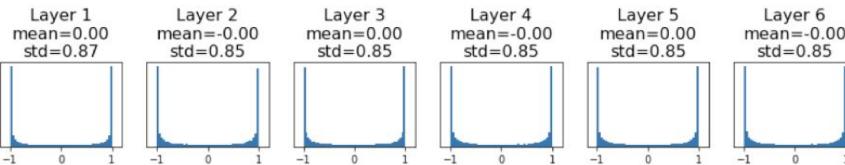
dims = [4096] * 7    Increase std of initial
hs = []
weights from 0.01 to 0.05
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.05 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)

```

All activations saturate

Q: What do the gradients look like?

A: Local gradients all zero, no learning = (



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 62

April 25, 2023

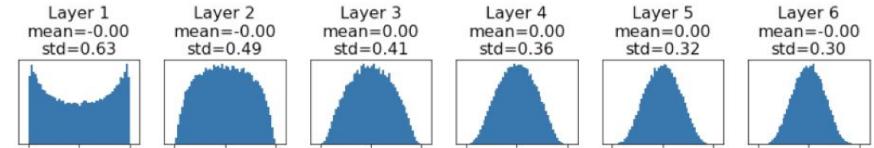
Weight Initialization: “Xavier” Initialization

```

dims = [4096] * 7    "Xavier" initialization:
hs = []                std = 1/sqrt(Din)
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.tanh(x.dot(W))
    hs.append(x)

```

“Just right”: Activations are nicely scaled for all layers!



Glorot and Bengio, “Understanding the difficulty of training deep feedforward neural networks”, AISTAT 2010

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 64

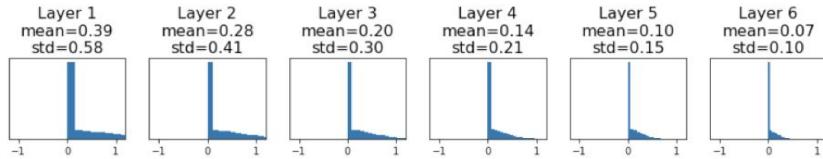
April 25, 2023

Weight Initialization: What about ReLU?

```

dims = [4096] * 7    Change from tanh to ReLU
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.maximum(0, x.dot(W))
    hs.append(x)

```



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 74

April 25, 2023

Xavier assumes zero centered activation function

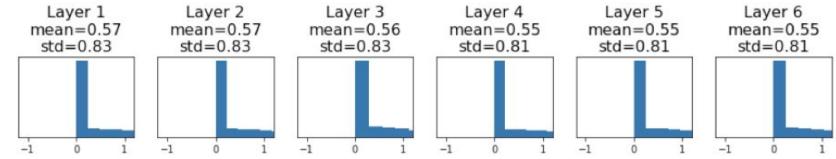
Activations collapse to zero again, no learning =()

Weight Initialization: Kaiming / MSRA Initialization

```

dims = [4096] * 7  ReLU correction: std = sqrt(2 / Din)
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) * np.sqrt(2/Din)
    x = np.maximum(0, x.dot(W))
    hs.append(x)

```



"Just right": Activations are nicely scaled for all layers!

He et al, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", ICCV 2015

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 75

April 25, 2023

Proper initialization is an active area of research...

Understanding the difficulty of training deep feedforward neural networks
by Glorot and Bengio, 2010

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks by Saxe et al, 2013

Random walk initialization for training very deep feedforward networks by Sussillo and Abbott, 2014

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification by He et al., 2015

Data-dependent Initializations of Convolutional Neural Networks by Krähenbühl et al., 2015

All you need is a good init, Mishkin and Matas, 2015

Fixup Initialization: Residual Learning Without Normalization, Zhang et al, 2019

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, Frankle and Carbin, 2019

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 76

April 25, 2023

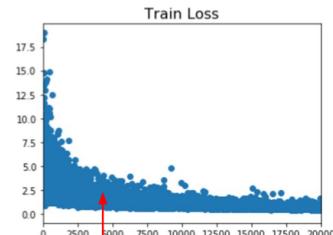
Training vs. Testing Error

Fei-Fei Li, Yunzhu Li, Ruohan Gao

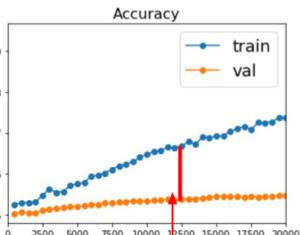
Lecture 7 - 77

April 25, 2023

Beyond Training Error

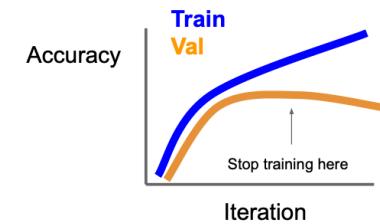
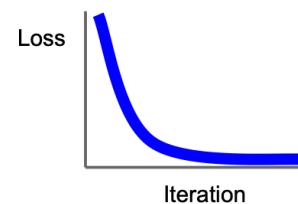


Better optimization algorithms help reduce training loss



But we really care about error on new data - how to reduce the gap?

Early Stopping: Always do this



Stop training the model when accuracy on the validation set decreases
Or train for a long time, but always keep track of the model snapshot
that worked best on val

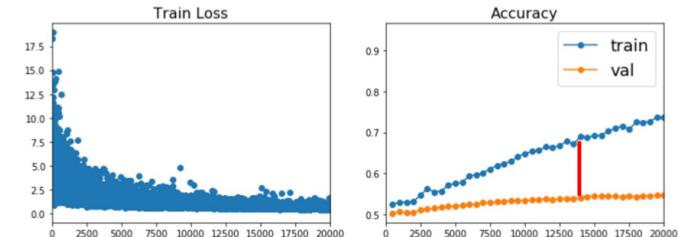
Model Ensembles

1. Train multiple independent models
2. At test time average their results

(Take average of predicted probability distributions, then choose argmax)

Enjoy 2% extra performance

How to improve single-model performance?



Regularization

Regularization: Add term to loss

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \boxed{\lambda R(W)}$$

In common use:

L2 regularization

$$R(W) = \sum_k \sum_l W_{k,l}^2 \quad (\text{Weight decay})$$

L1 regularization

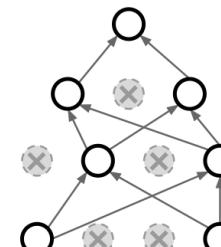
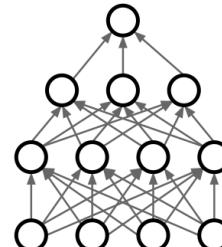
$$R(W) = \sum_k \sum_l |W_{k,l}|$$

Elastic net (L1 + L2)

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

Regularization: Dropout

In each forward pass, randomly set some neurons to zero
Probability of dropping is a hyperparameter; 0.5 is common



Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting", JMLR 2014

Regularization: Dropout

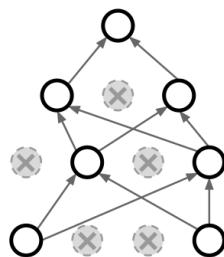
```
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
    """ X contains the data """

    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = np.random.rand(*H1.shape) < p # first dropout mask
    H1 *= U1 # drop!
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    U2 = np.random.rand(*H2.shape) < p # second dropout mask
    H2 *= U2 # drop!
    out = np.dot(W3, H2) + b3

    # backward pass: compute gradients... (not shown)
    # perform parameter update... (not shown)
```

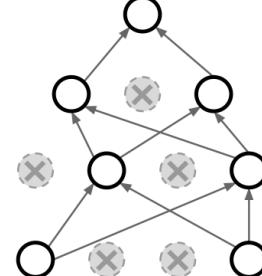
Example forward pass with a 3-layer network using dropout



Regularization: Dropout

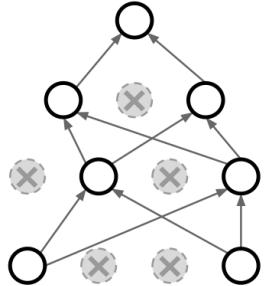
How can this possibly be a good idea?

Forces the network to have a redundant representation;
Prevents co-adaptation of features



Regularization: Dropout

How can this possibly be a good idea?



Another interpretation:

Dropout is training a large **ensemble** of models (that share parameters).

Each binary mask is one model

An FC layer with 4096 units has $2^{4096} \sim 10^{1233}$ possible masks!
Only $\sim 10^{82}$ atoms in the universe...

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

86

April 25, 2023

Dropout Summary

```
""" Vanilla Dropout: Not recommended implementation (see notes below) """
p = 0.5 # probability of keeping a unit active. higher = less dropout

def train_step(X):
    """ X contains the data """

    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = np.random.rand(*H1.shape) < p # first dropout mask
    H1 *= U1 # drop!
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    U2 = np.random.rand(*H2.shape) < p # second dropout mask
    H2 *= U2 # drop!
    out = np.dot(W3, H2) + b3

    # backward pass: compute gradients... (not shown)
    # perform parameter update... (not shown)

def predict(X):
    # ensembled forward pass
    H1 = np.maximum(0, np.dot(W1, X) + b1) * p # NOTE: scale the activations
    H2 = np.maximum(0, np.dot(W2, H1) + b2) * p # NOTE: scale the activations
    out = np.dot(W3, H2) + b3
```

drop in train time

scale at test time

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

93

April 25, 2023

More common: “Inverted dropout”

```
p = 0.5 # probability of keeping a unit active. higher = less dropout

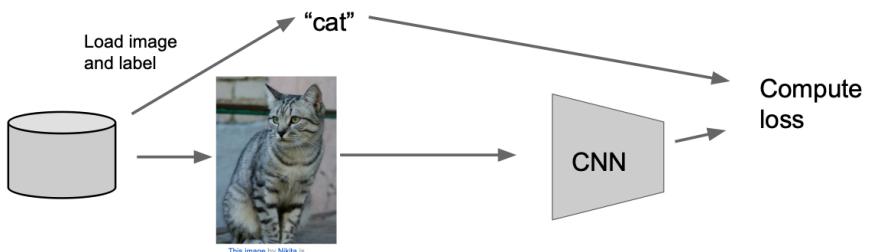
def train_step(X):
    # forward pass for example 3-layer neural network
    H1 = np.maximum(0, np.dot(W1, X) + b1)
    U1 = (np.random.rand(*H1.shape) < p) / p # first dropout mask. Notice /p!
    H1 *= U1 # drop!
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    U2 = (np.random.rand(*H2.shape) < p) / p # second dropout mask. Notice /p!
    H2 *= U2 # drop!
    out = np.dot(W3, H2) + b3

    # backward pass: compute gradients... (not shown)
    # perform parameter update... (not shown)

def predict(X):
    # ensembled forward pass
    H1 = np.maximum(0, np.dot(W1, X) + b1) # no scaling necessary
    H2 = np.maximum(0, np.dot(W2, H1) + b2)
    out = np.dot(W3, H2) + b3
```

test time is unchanged!

Regularization: Data Augmentation



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

94

April 25, 2023

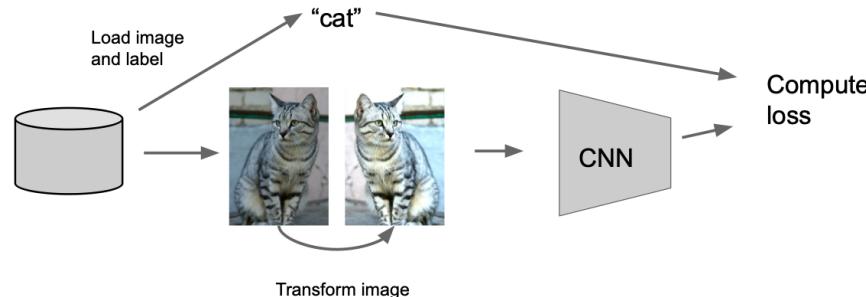
Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

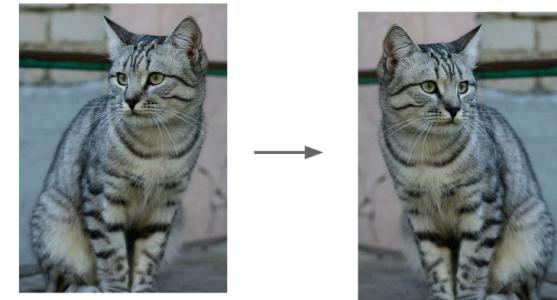
97

April 25, 2023

Regularization: Data Augmentation



Data Augmentation Horizontal Flips

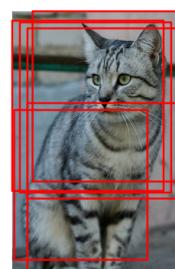


Data Augmentation Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch

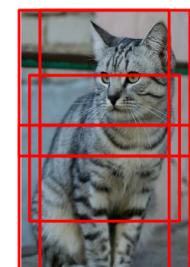


Data Augmentation Random crops and scales

Training: sample random crops / scales

ResNet:

1. Pick random L in range [256, 480]
2. Resize training image, short side = L
3. Sample random 224 x 224 patch



Testing: average a fixed set of crops

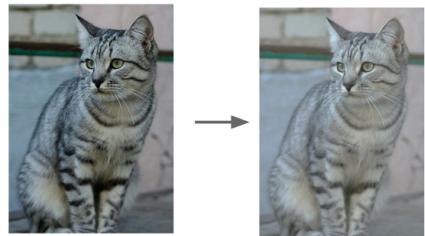
ResNet:

1. Resize image at 5 scales: {224, 256, 384, 480, 640}
2. For each size, use 10 224 x 224 crops: 4 corners + center, + flips

Data Augmentation

Color Jitter

Simple: Randomize contrast and brightness



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

102

April 25, 2023

Data Augmentation

Color Jitter

Simple: Randomize contrast and brightness



More Complex:

1. Apply PCA to all [R, G, B] pixels in training set
2. Sample a “color offset” along principal component directions
3. Add offset to all pixels of a training image

(As seen in [Krizhevsky et al. 2012], ResNet, etc)

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

103

April 25, 2023

Data Augmentation

Get creative for your problem!

Examples of data augmentations:

- translation
- rotation
- stretching
- shearing,
- lens distortions, ... (go crazy)

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

104

April 25, 2023

Choosing Hyperparameters

(without tons of GPUs)

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

113

April 25, 2023

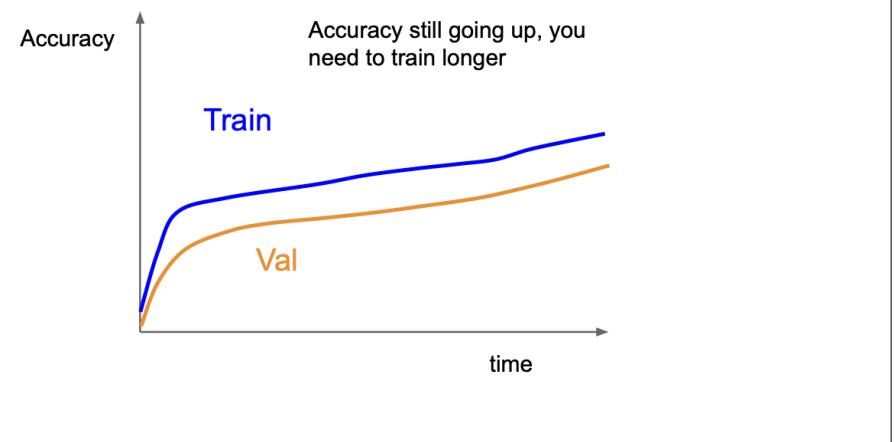
Choosing Hyperparameters

- Step 1:** Check initial loss
- Step 2:** Overfit a small sample
- Step 3:** Find LR that makes loss go down
- Step 4:** Coarse grid, train for ~1-5 epochs
- Step 5:** Refine grid, train longer
- Step 6:** Look at loss and accuracy curves

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 -

119 April 25, 2023

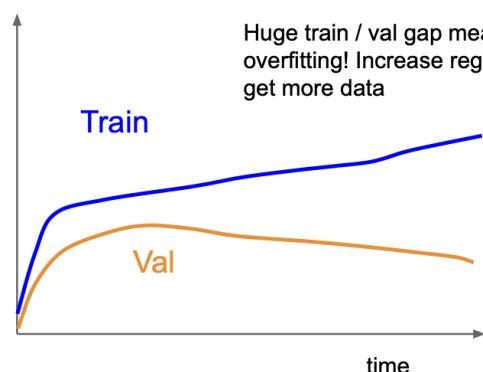


Fei-Fei Li, Yunzhu Li, Ruohan Gao

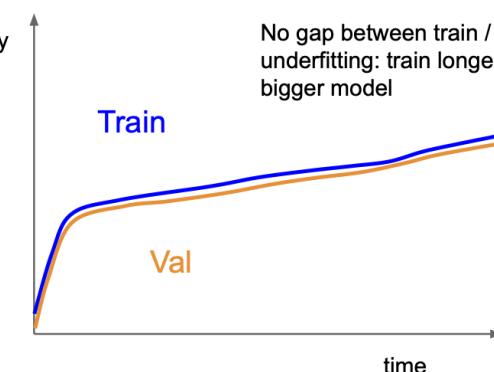
Lecture 7 -

120 April 25, 2023

Huge train / val gap means overfitting! Increase regularization, get more data



No gap between train / val means underfitting: train longer, use a bigger model



Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 121

April 25, 2023

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 7 - 122

April 25, 2023

Cross-validation

We develop "command centers" to visualize all our models training with different hyperparameters

check out [weights and biases](#)



Summary

- Improve your training error:
 - Optimizers
 - Learning rate schedules
- Improve your test error:
 - Regularization
 - Choosing Hyperparameters