Neural Network Optimization 1

CS 519: Deep Learning, Winter 2018

Fuxin Li

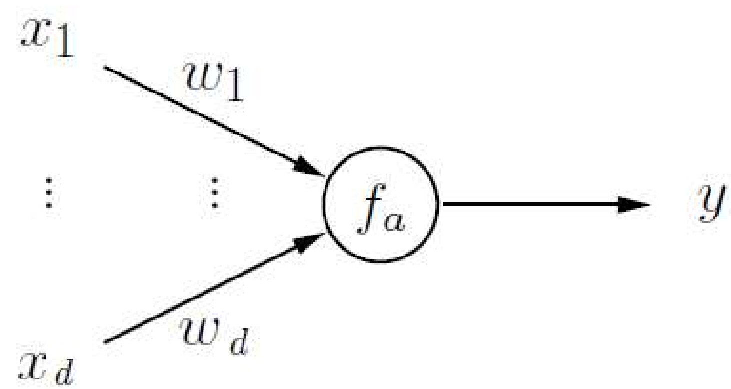
With materials from Zsolt Kira

Backpropagation learning of a network

- The algorithm
  - 1. Compute a forward pass on the compute graph (DAG) from the input to all the outputs
  - 2. Backpropagate all the outputs back all the way to the input and collect all gradients $\mathbf{G}$
  - 3. $\mathbf{W} = \mathbf{W} - \alpha\mathbf{G}$ for all the weights in all layers

Modules (Layers)

- Each layer can be seen as a module
- Given input, return
  - Output $f_a(x)$
  - Network gradient $\frac{\partial f_a}{\partial x}$
  - Gradient of module parameters $\frac{\partial f_a}{\partial w_a}$
- During backprop, propagate/update
  - Backpropagated gradient $\frac{\partial E}{\partial f_a}$



$$\frac{\partial E}{\partial \boldsymbol{W}_k} = \frac{\partial E}{\partial f_k} g\big(f_{k-1}(x)\big) = \frac{\partial E}{\partial f_{k+1}} \frac{\partial f_{k+1}}{\partial f_k} g(f_{k-1}(x))$$

where $f_k(x) = w_k^\top g\big(f_{k-1}(x)\big), f_0(x) = x$

The abundance of online layers

Core Layers

- Layer
- Methods
- Masking
- TimeDistributedMerge
- Merge
- Dropout
- Activation
- Reshape
- Permute
- Flatten
- RepeatVector
- Dense
- ActivityRegularization
- TimeDistributedDense
- AutoEncoder
- MaxoutDense
- Lambda
- LambdaMerge
- Siamese
- Highway

Convolutional Layers

- Convolution1D
- Convolution2D
- MaxPooling1D
- AveragePooling1D
- MaxPooling2D
- AveragePooling2D
- UpSampling1D
- UpSampling2D
- ZeroPadding1D
- ZeroPadding2D

Recurrent Layers

- Recurrent
- SimpleRNN
- GRU
- LSTM

Advanced Activations Layers

- LeakyReLU
- PReLU
- ELU
- ParametricSoftplus
- ThresholdedLinear
- ThresholdedReLU

Normalization Layers

- BatchNormalization

Embedding Layers

- Embedding

Noise layers

- GaussianNoise
- GaussianDropout

Learning Rates

- Gradient descent is only guaranteed to converge with *small enough* learning rates
  - So that's a sign you should decrease your learning rate if it explodes
- Example:
  - $C(w) = \frac{1}{2}w^2$
  - Learning rate of $\alpha = 0.5$
  - $\alpha = 1$
  - $\alpha = 2$
  - $\alpha = 3$

$w_0 = 1$

$w_1 = 1 - 0.5 \cdot 1$

$= 0.5$

$w_2 = 0.5 - 0.5 \cdot 0.5$

$= 0.25$

Weight decay regularization

- Instead of using a normal step, add a

$$\mathbf{G} = \mathbf{G} + \lambda \mathbf{W}$$

- This corresponds to:

$$\min_{\mathbf{W}} \frac{1}{N} \sum_{i=1}^{N} l(f(x_i; \mathbf{W}), y_i) + \frac{1}{2} \lambda \|\mathbf{W}\|^2$$

- Early stopping as well!
    - Help generalization

$W = W - \alpha G$

$W = W - \alpha G - \alpha\lambda W$

$\alpha\lambda = \beta$

Momentum

$(D_1 = -\alpha G_0)\ W_1 = W_0 - \alpha G_0$

$D_2 = -\alpha\mu G_0 - \alpha G_1$

$W_2 = W_1 - \alpha\mu G_0 - \alpha G_1$

$W_3 = W_2 - \alpha\mu^2 G_0 - \alpha\mu G_1 - \alpha G_2$

- Basic updating equation (with momentum):

$$D_0 = 0$$
$$D_{t+1} = \mu D_t - \alpha G_t$$
$$W_{t+1} = W_t + D_{t+1}$$

  - $\mu = 0.6 \sim 0.9$, a lot of "inertia" in optimization

- Check the previous example with a momentum of 0.5

$\min \frac{1}{2} w^2$

$\alpha = 0.5$

$W_0 = 1$

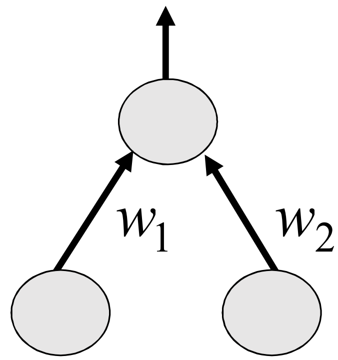
$W_1 = 0.5$

$W_2 = 0.20$

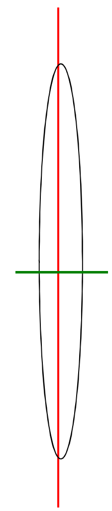
Normalization

- Each component to 0 mean, 1 standard deviation
- For ease of L2 regularization + optimization convergence rates

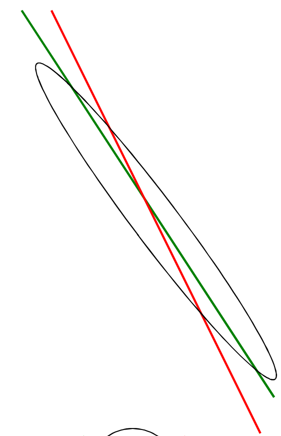
color indicates training case



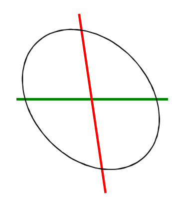
0.1, 10
0.1, -10



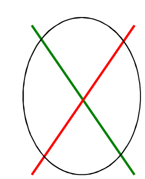
101, 101
101, 99



1, 1
1, -1



1, 1
1, -1



Computing the energy function and gradient

- Usual ERM energy function

$$\min_W E(f) = \sum_{i=1}^{n} L(f(x_i; W), y_i)$$

$$\nabla_W E = \sum_{i=1}^{n} \frac{\partial L(f(x_i; W), y_i)}{\partial W}$$

- One problem:
  - Very slow to compute when n is large
  - One gradient step takes long time!
  - Approximate?

Stochastic Mini-batch Approximation

$$\min_W E(f) = \sum_{i=1}^{n} L(f(x_i; W), y_i)$$

$$\nabla_W E = \sum_{i=1}^{n} \frac{\partial L(f(x_i; W), y_i)}{\partial W}$$

$$\nabla_W \tilde{E} \approx \sum_{i \in N_m} \frac{\partial L(f(x_i; W), y_i)}{\partial W} \qquad N_m \subset \{1, \dots, n\}$$

$$E_x\left[\frac{\partial L(f(x_i; W), y_i)}{\partial W}\right]$$

- Ensure the expectation is the same

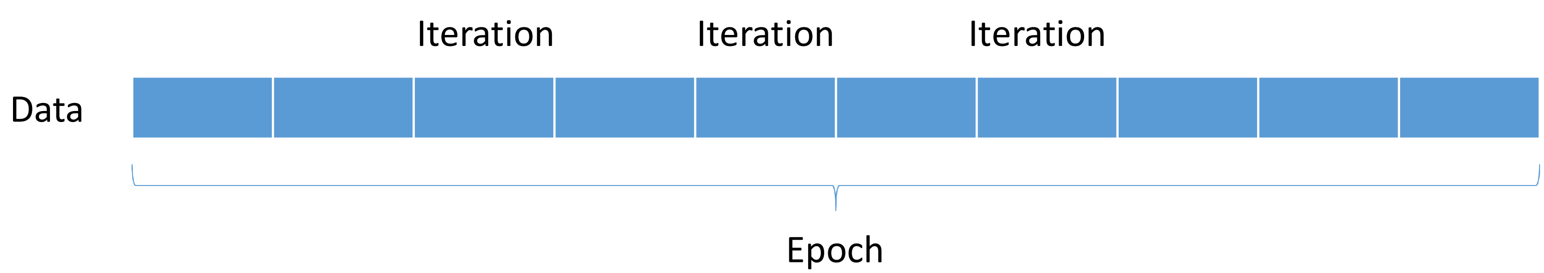
$$\mathbb{E}(\nabla_W \tilde{E}) = \nabla_W E$$

  - Uniformly sample every time
    - Sample how many? 1 (SGD) – 256 (Mini-batch SGD)
    - Common mini-batch size is 32-256
    - In practice: dependent on GPU memory size

In Practice

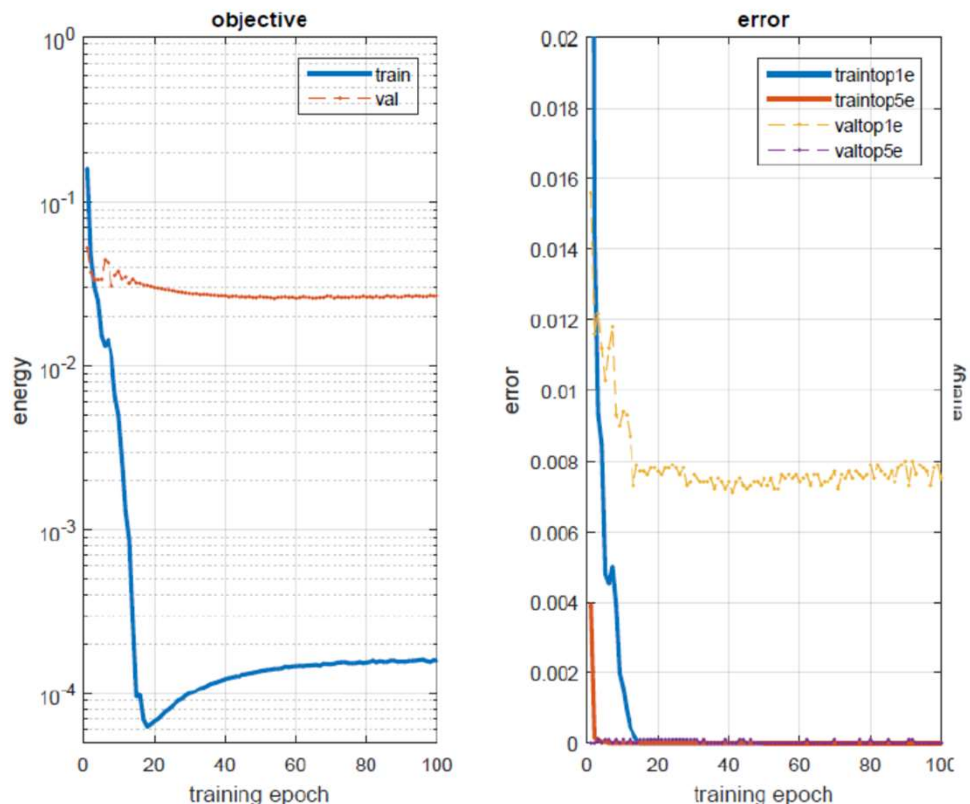
(n-2 3 5 7)
(8 9 2 1)

- Randomly re-arrange the input examples
- Use a fixed order on the input examples
- Define an *iteration* to be every time the gradient is computed
- An *epoch* to be every time that all the input examples is looped through once



A practical run of training a neural network

- Check:
  - Energy
  - Training error
  - Validation error



M = 64

Data Augmentation

- Create artificial data to increase the size of the dataset
- Example: Elastic deformations
  on MNIST

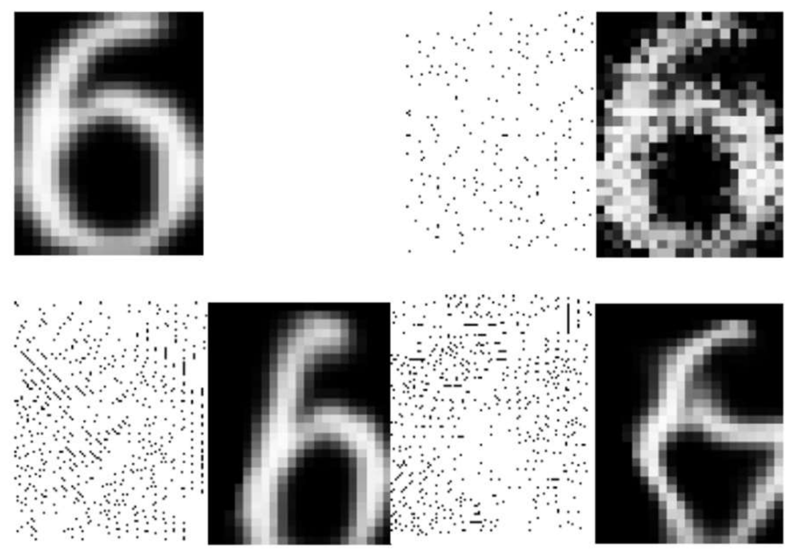
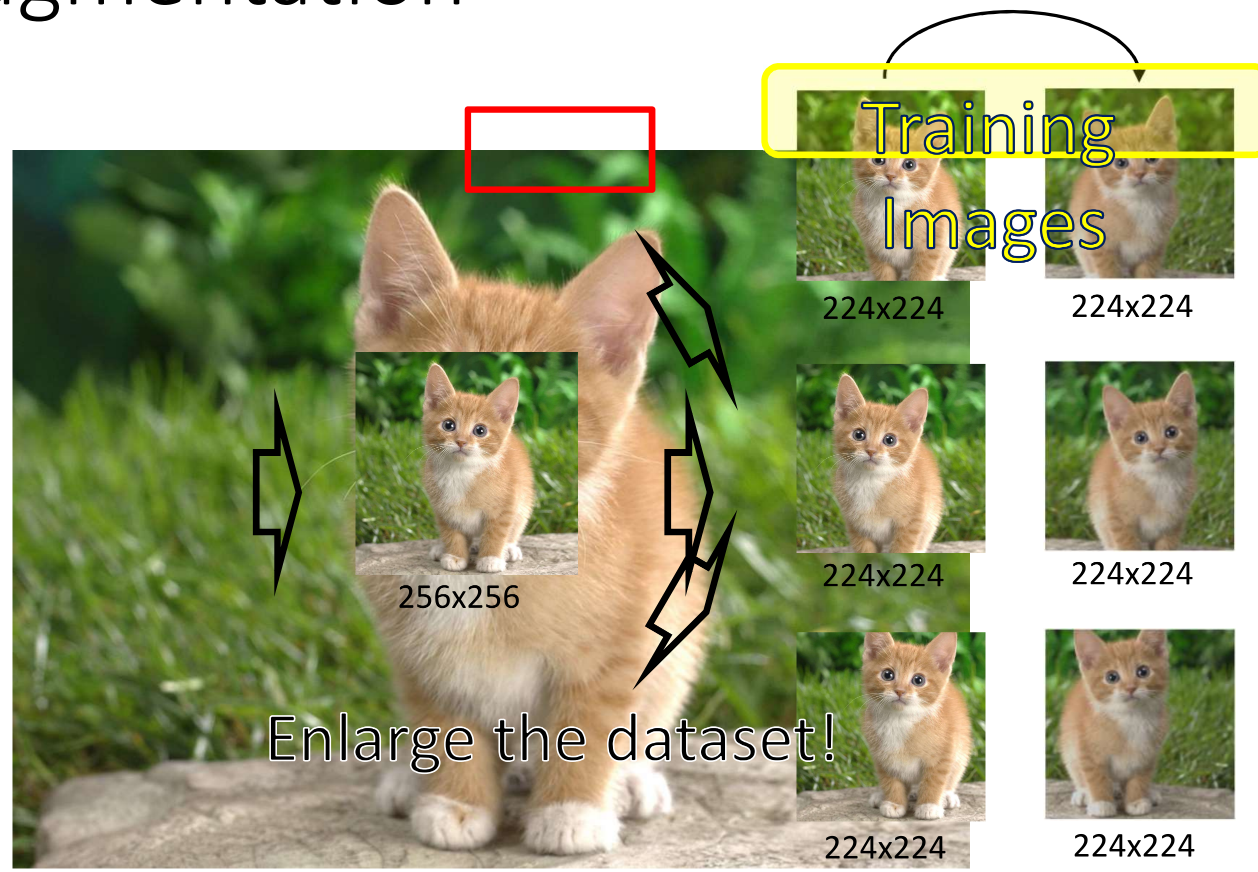


Figure 2. Top left: Original image. Right and bottom: Pairs of displacement fields with various smoothing, and resulting images when displacement fields are applied to the original image.

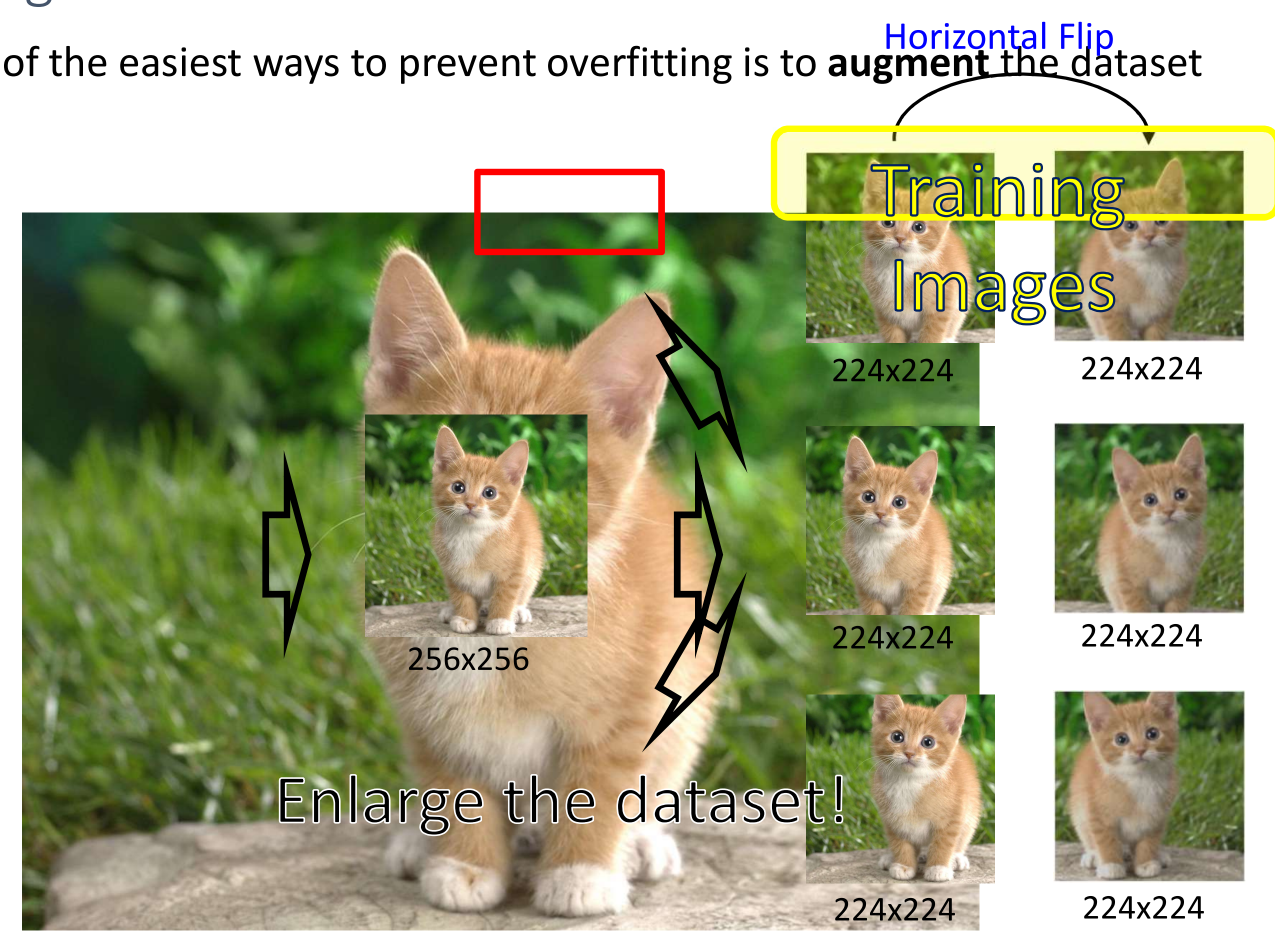
Data Augmentation



Training Image

Data Augmentation

- One of the easiest ways to prevent overfitting is to **augment** the dataset



Training Image

CIFAR-10 dataset

- 60,000 images in 10 classes
  - 50,000 training
  - 10,000 test
- Designed to mimic MNIST
- 32x32
- Assignment (will post on Canvas with more explicity):
  - Write your own backpropagation NN to test on CIFAR-10