Albert Einstein: Insanity Is Doing the Same Thing Over and Over Again and Expecting Different Results

Machine learning:

Organizational

- Homework 3 released → Due next Wednesday
- Homework 4 will be released then (you'll have 2 weeks)

→ Homework can be perceived as difficult! Start early!

- Debugging "flipped classroom"

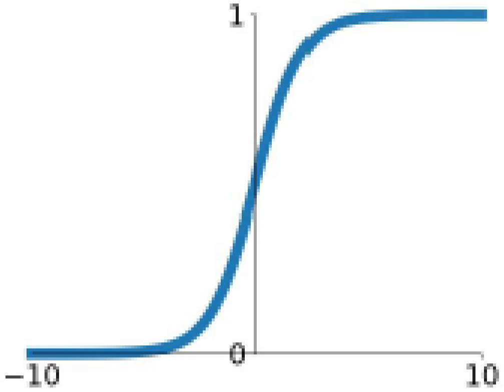
- Start thinking about project teams! **Groups of 4**. Not 3, not 5.
- Proposals due in April

CNNs – General Layout

- **One-time setup**
  - **Architecture (Lecture 10)**
  - **Activation functions (sigmoid, ReLU, …)**
  - **Regularization (batch norm, dropout)**

- Training
  - Data collection: Preprocessing, Augmentation
  - Training via SGD (update rules)

Sigmoid



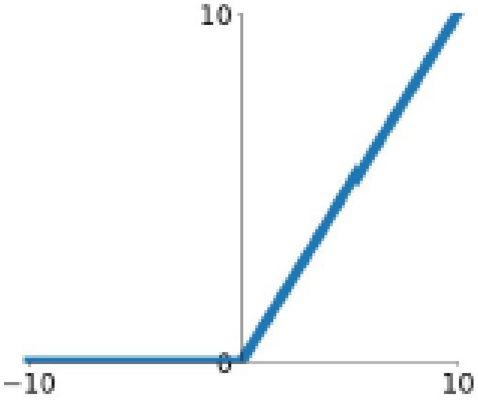
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

- Squashes input to [0,1]
- Historically popular:
  Saturating firing rate of a neuron

Problems

- Gradient: $\frac{\partial \sigma(x)}{\partial x} = (1 - \sigma(x))\sigma(x)$
  → Gradient vanishes for saturated neurons
- Outputs are not zero-centered

Rectified Linear Unit



ReLU $\mathrm{ReLU}(x) = \max(0, x)$

- No saturation in positive regime
- Computationally efficient
- Converges much faster than previous func.s
- Closer to biological neuron activation

Problems

- Again not zero-centered!
- May permanently de-activate

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. NeurIPS (pp. 1097-1105).

Weight Initialization

- Initialization is an active field of research
  (in neural networks and beyond, e.g. image registration)

- Xavier and He initialization played an important role in the success of DL

- If you are using ReLU as recommended: He initialization is your friend!

Preprocessing

- Zero-center data
- Try normalizing images

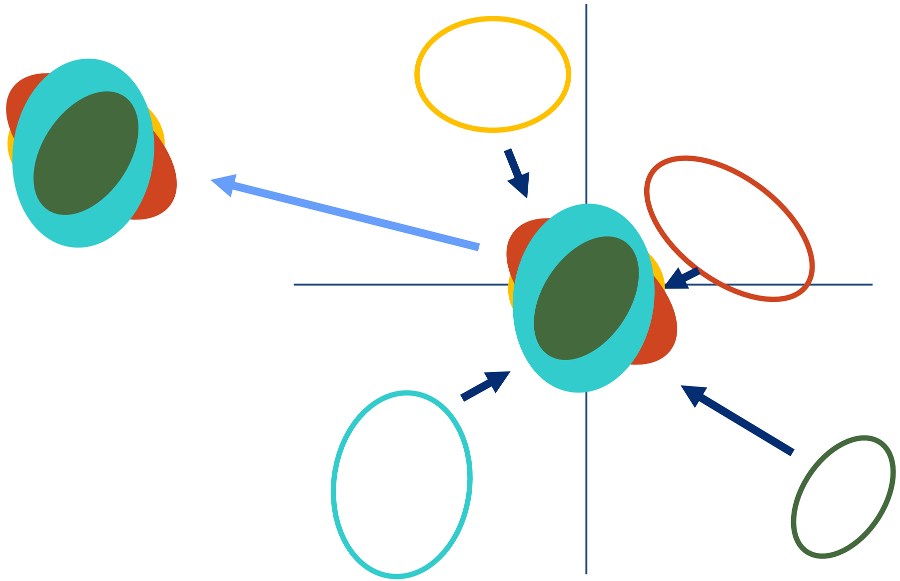
- Do not (necessarily) consider decorrelation, whitening or other techniques for images, but this may be different for other input data

At inference time:

Apply the same transformation (e.g. mean subtraction) with values extracted from the training data.

Move Batches to Standard Location

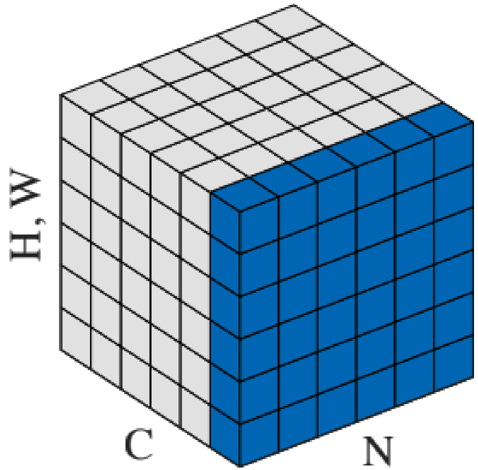
Eliminate covariate shift by “moving” batches to zero mean and unit standard dev



→ Then, move entire collection to desirable location: **Batch normalization**

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.

Batch Normalization



1. Compute empirical mean and variance for each channel
$$E[x^{(k)}], \mathrm{Var}[x^{(k)}]$$

2. Normalize to unit Gaussian
$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

3. Squash output to beneficial range
$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}$$

These are parameters and are learned during training.

Network can learn identity!
$$\gamma^{(k)} = \mathrm{Var}[x^{(k)}]$$
$$\beta^{(k)} = E[x^{(k)}]$$

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.

Dropout in Forward Pass

Without dropout:

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}), \end{aligned}$$

With dropout:

$$\begin{aligned} r_j^{(l)} &\sim \mathrm{Bernoulli}(p), \\ \widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^l + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}). \end{aligned}$$

1. For every node j and layer l, determine Bernoulli number {0,1}
2. Drop outputs
3. ???
4. Profit.

CNNs – General Layout

- One-time setup
  - Architecture (Lecture 10)
  - Activation functions (sigmoid, ReLU, …)
  - Regularization (batch norm, dropout)

- **Training**
  - **Data collection: Preprocessing, Augmentation**
  - **Training via SGD (update rules)**

Adding Momentum $W' = \arg\min_W L(W)$

SGD

$$W_{t+1} = W_t - \alpha \nabla_W L(W_t)$$

- Update in negative gradient direction

Problems of SGD?

Adding Momentum $W' = \arg\min_W L(W)$

SGD

$$W_{t+1} = W_t - \alpha \nabla_W L(W_t)$$

- Update in negative gradient direction

SGD + Momentum

$$v_{t+1} = \rho v_t + \alpha \nabla_W L(W_t)$$
$$W_{t+1} = W_t - v_{t+1}$$

- Replace gradient with *velocity*
- Velocity: Running mean of gradients
- ρ determines friction (ρ > 0.9)
- Update in negative velocity direction

This simple strategy helps in all previous problems!

Enter RMSProp

RMSProp is **unpublished** but has similarities to ADADELTA:
Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

$$S_i = S_i + \sqrt{(g_t)_i^2}$$

$$g_t = \nabla_W L(W_t)$$

$$S_i = \rho \cdot S_i + (1-\rho)(g_t)_i^2 \quad \text{with } S_i(t=0) = 0$$

$$(\mathrm{d}W_t)_i = \frac{\alpha}{\sqrt{S_i} + \epsilon}(g_t)_i$$

$$\boxed{W_{t+1} = W_t - \mathrm{d}W_t}$$

1. Compute gradient
2. Compute “’discounted” element-wise squared gradient
3. Compute gradient update with **parameter-wise** learning rate
4. Apply gradient update

Adam

[Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.]

$$g_t = \nabla_W L(W_t)$$

Bias correction

$$S_i^{(1)} = \left(\rho_1 S_i^{(1)} + (1-\rho_1)(g_t)_i\right)\boxed{(1-\rho_1^t)^{(-1)}}$$

$$S_i^{(2)} = \left(\rho_2 S_i^{(2)} + (1-\rho_2)(g_t)_i^2\right)\boxed{(1-\rho_2^t)^{(-1)}}$$

$$(\mathrm{d}W_t)_i = \frac{\alpha}{\sqrt{S_i^{(2)}} + \epsilon} S_i^{(1)}$$

$$W_{t+1} = W_t - \mathrm{d}W_t$$

$$\boxed{\begin{aligned} S_i^{(1)}(t=0) &= 0 \\ S_i^{(2)}(t=0) &= 0 \end{aligned}}$$

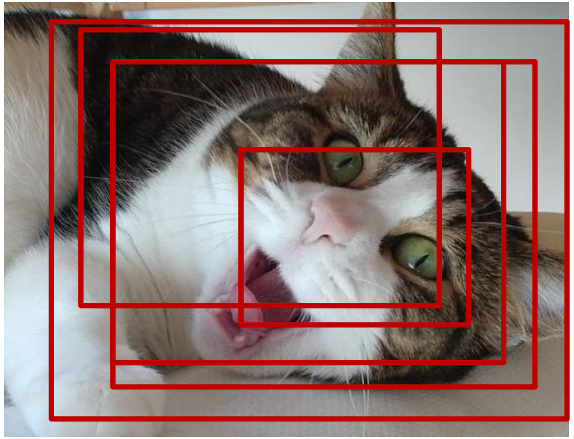
1. Compute gradient
2. Compute first momentum (“velocity”)
3. Compute second momentum (parameter-wise normalization)
4. Compute update with **momentum** and **parameter-wise** learning rate
5. Apply update

Image Transformations to Use for Augmentation

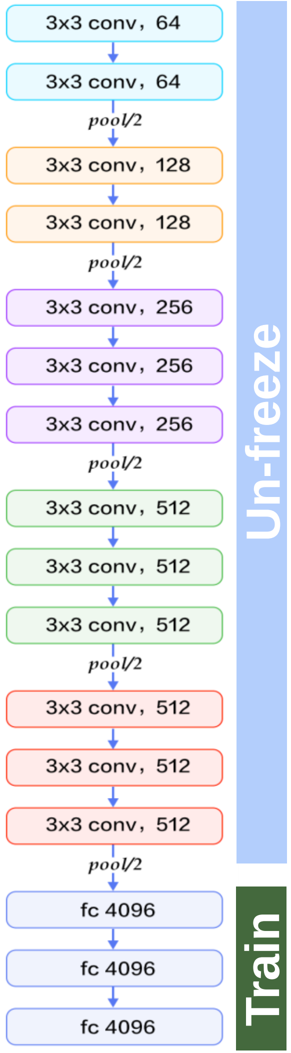
Rule of thumb
Every transformation that yields a **valid** image.

Examples: All these are random (within reasonable ranges)

- Horizontal / vertical flips
- Rotations and translations
- Noise (!)
- Scaling
- Cropping
- Color variations
- Distortions



→ We will see an interesting example of this soon!



Lower learning rate!
E.g. 1/10 of LR

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. CVPR (pp. 806-813).

Transfer learning:
Slightly more data

- Set-up network architecture
- Initialize last layers randomly
- Train new parameters

Second step: After some improvement in training

- Finetune complete network
- Carefully adjust LR to avoid “**forgetting**”

Q: Why does this work?