Practical Deep Learning

Naeemullah Khan

naeemullah.khan@kaust.edu.sa

KAUST Academy
King Abdullah University of Science and Technology

February 9, 2024

- Practical Implementation of Deep Learning algorithms is just as much an art as it is a science.
- The main takeaways if not to start from scratch rather to build on top of the previous knowledge.
- Today, we will look at some important tools used in the practical implementation of Deep Learning algorithms.

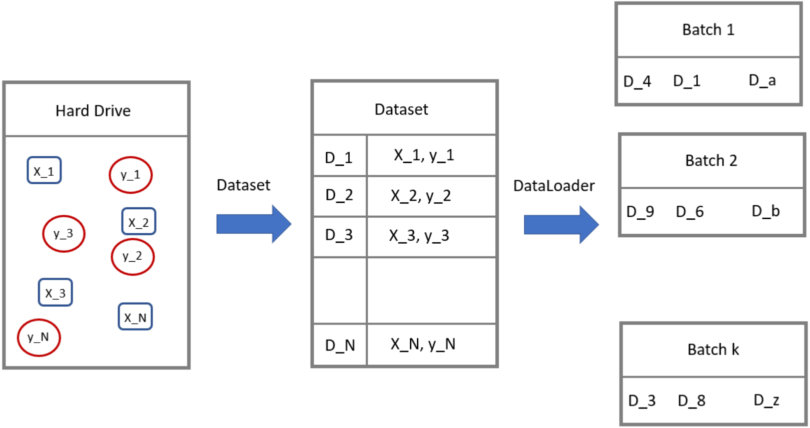
Outline

- Data Handling
- Data Augmentation
- Transfer Learning
- Ensembling
- Dropout
- Batch Normalization

DataLoaders

- As we have previously established that Deep Learning has been made possible by large amount of data and computational resourse
- An important aspect to keep in mind is the data handling:
  - How do we handle large amounts of data?
  - How to we read different components of data (from possible different parts of our hard drive) and provide it to our training algorithms?
  - How do we feed this data to SGD algorthims in a streamlined manner?
- PyTorch provides Dataset and DataLoaders to handle data in an efficient manner.
- We will extend the Dataset and DataLoaders class to construct our own Dataloaders

DataLoaders (cont.)



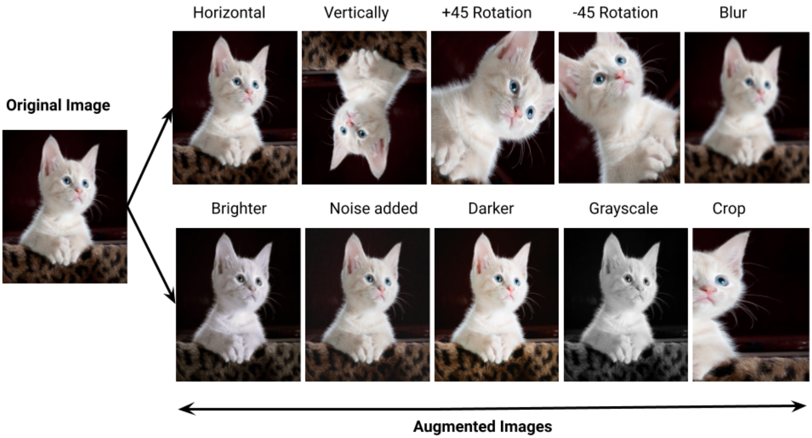
Data Augmentation

- Data is the fundamental building block of any machine learning algorithm
- In several applications we don't have access to unlimited data
- So we use Data Augmentation techniques to improve the preformance of our models
- Note: It is better to spend time on data rather than fine-scale architecture search in deep learning

Data Augmentation (cont.)

- Create virtual training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion
  - Translation
  - Rotation

Data Augmentation (cont.)

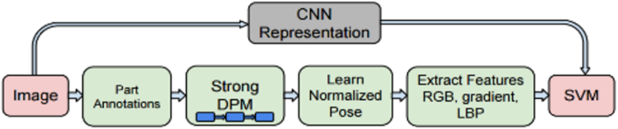


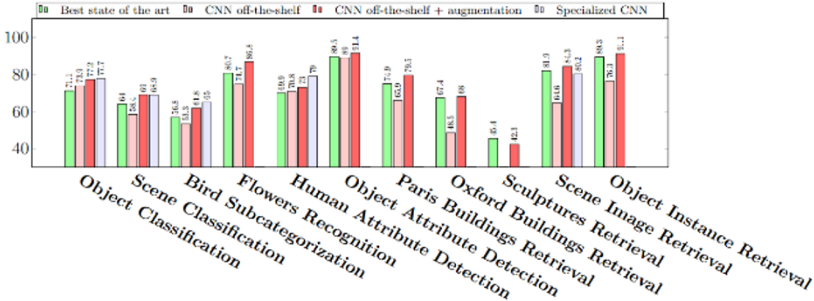
[0] https://pranjal-ostwal.medium.com/data-augmentation-for-computer-vision-b88b818b6010

- Improvement of learning in a **new** task through the transfer of knowledge from a **related** task that has already been learned.
- We will look at one strategy of transfer learning called Fine-Tuning

When to fine-tune your model?

- New dataset is small with distribution similar to original dataset.
  - Keep the feature extraction part fixed and fine-tune the classifier part of the network
- New dataset is large with similar distribution to the original dataset
  - Fine tune both the feature extractor and the classifier part of the network
- New dataset is small but different distribution from the original dataset
  - Use SVM classifier on the features extracted from the feature extractor part of the Network
- New dataset is large and different distribution from the original dataset
  - Fine tune both the feature extractor and the classifier part of the network





[0] Razavian et al. 2014

Finetuning

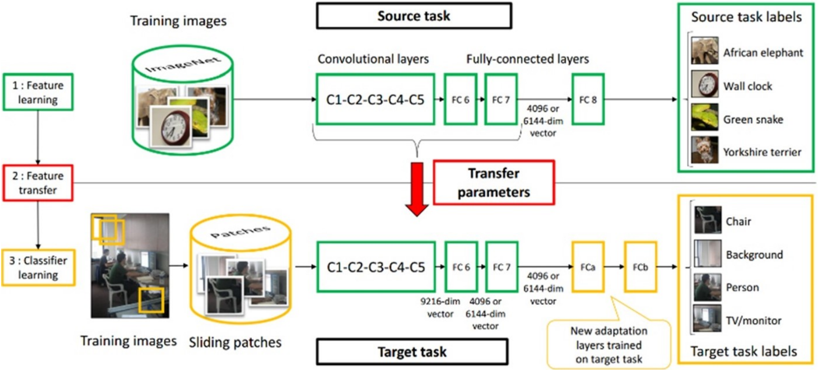


Figure 2: Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks

[0] Oquab et al. CVPR 2014

Ensembling

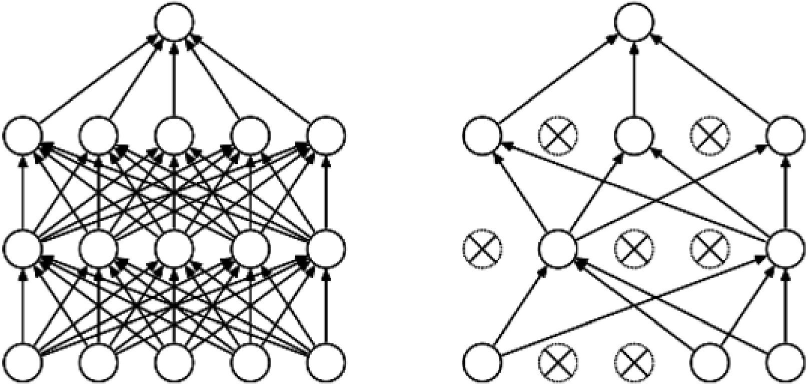
- Team-work is the best policy
- Multiple networks for the same task
- Max Voting for final classification

Ensembling - A simple Analysis

- Let's assume that we have a test dataset with N elements and an ensemble of M models.
- Also assume that the probability of error of the label for an image on a model in the ensemble is denoted by $p(e)$ and is i.i.d
- For an example assume $M = 3$ and $e = 0.01$
- Then probability of error of label for the max voting ensemble will be

$$p(e) = 1 - (1-e)^3 - \binom{3}{2}(1-e)^2 e$$

- For the above example $p(e) = 0.0003$, which is significantly lower than a single model

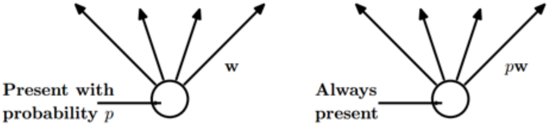


(a) Standard Neural Net

(b) After applying dropout.

[0] Srivastava JMLR 2014

Dropout



(a) At training time (b) At test time

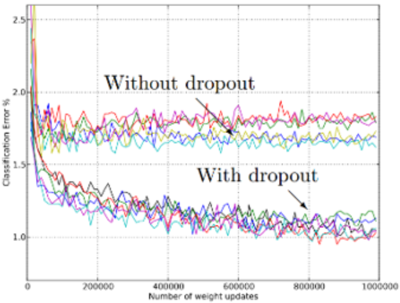
Intuition: successful conspiracies

- 50 people planning a conspiracy
- Strategy A: plan a big conspiracy involving 50 people
  - Likely to fail. 50 people need to play their parts correctly.
- Strategy B: plan 10 conspiracies each involving 5 people
  - Likely to succeed!

Main Idea: approximately combining exponentially many different neural network architectures efficiently

[0] Srivastava JMLR 2014

Dropout



Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

[0] Srivastava JMLR 2014

Batch Normalization

- Consider a single layer $y = Wx$
- The following could lead to tough optimazation
  - Inputs x are not centered around zero (need large bias)
  - Inputs x have different scaling per element (entries in W will need to vary a lot)

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Batch Normalization

- Consider a single layer $y = Wx$
- The following could lead to tough optimazation
  - Inputs x are not centered around zero (need large bias)
  - Inputs x have different scaling per element (entries in W will need to vary a lot)
- **Idea:** Force inputs to be "nicely scaled" at each layer!

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Batch Normalization

- Consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Batch Normalization

- Consider a batch of activations at some layer. To make each dimension zero-mean unit-variance, apply:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

- **Problem:** What if zero-mean, unit variance is too hard of a constraint?

[0]Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Batch Normalization

Input: $x : N \times D$

Learnable scale and shift parameters:

$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$ Per-channel mean, shape is D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$$ Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$ Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$ Output, Shape is N x D

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Estimates depend on minibatch;
can't do this at test-time!

Input: $x : N \times D$

Learnable scale and shift parameters:

$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$$\boxed{\begin{aligned} \mu_j &= \frac{1}{N}\sum_{i=1}^{N} x_{i,j} && \text{Per-channel mean, shape is D} \\ \sigma_j^2 &= \frac{1}{N}\sum_{i=1}^{N} (x_{i,j} - \mu_j)^2 && \text{Per-channel var, shape is D} \end{aligned}}$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Normalized x,
Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output,
Shape is N x D

Input: $x : N \times D$

Learnable scale and shift parameters:

$$\gamma, \beta : D$$

During testing batchnorm becomes a linear operator!
Can be fused with the previous fully-connected or conv layer

$\mu_j =$ (Running) average of values seen during training — Per-channel mean, shape is D

$\sigma_j^2 =$ (Running) average of values seen during training — Per-channel var, shape is D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

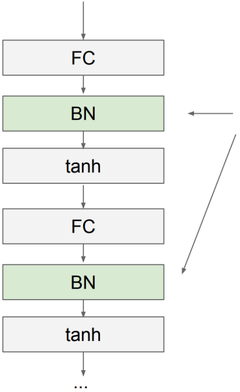
Normalized x, Shape is N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Output, Shape is N x D

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

Batch Normalization



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

Batch Normalization for **fully-connected** networks

$$x: N \times D$$

Normalize ↓ (over N)

$$\mu, \sigma: 1 \times D$$
$$\gamma, \beta: 1 \times D$$
$$y = \gamma (x - \mu) / \sigma + \beta$$

Batch Normalization for **convolutional** networks (Spatial Batchnorm, BatchNorm2D)

$$x: N \times C \times H \times W$$

Normalize ↓ (over N, H, W)

$$\mu, \sigma: 1 \times C \times 1 \times 1$$
$$\gamma, \beta: 1 \times C \times 1 \times 1$$
$$y = \gamma (x - \mu) / \sigma + \beta$$

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

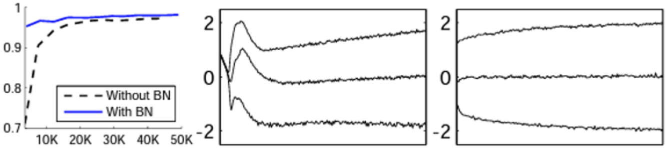
Batch Normalization

- Advantages:
  - Makes deep networks much easier to train!
  - Improves gradient flow
  - Allows higher learning rates, faster convergence
  - Networks become more robust to initialization
  - Acts as regularization during training
  - Zero overhead at test-time: can be fused with conv!

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao

- Advantages:
  - Makes deep networks much easier to train!
  - Improves gradient flow
  - Allows higher learning rates, faster convergence
  - Networks become more robust to initialization
  - Acts as regularization during training
  - Zero overhead at test-time: can be fused with conv!
- Disadvantages:
  - Behaves differently during training and testing: this is a very common source of bugs!

[0] Slide based on CS231n by Fei-Fei Li, Yunzhu Li & Ruohan Gao



(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

[0] Ioffe and Szegedy 2015

Things to Remember

- Training Deep Networks
  - Dropout
  - Data augmentation
  - Activation
  - Batch normalization
- Transfer learning
  - Use Fine-tuning when possible

Full Deep Learning Pipeline

- Data Pre-processing
- Architecture
- Loss
- Optimizer
- DataLoaders
- Data Augmentation
- Fine-Tuning
- Ensembling