Working With Data

EE599 Deep Learning

Keith M. Chugg
Spring 2020

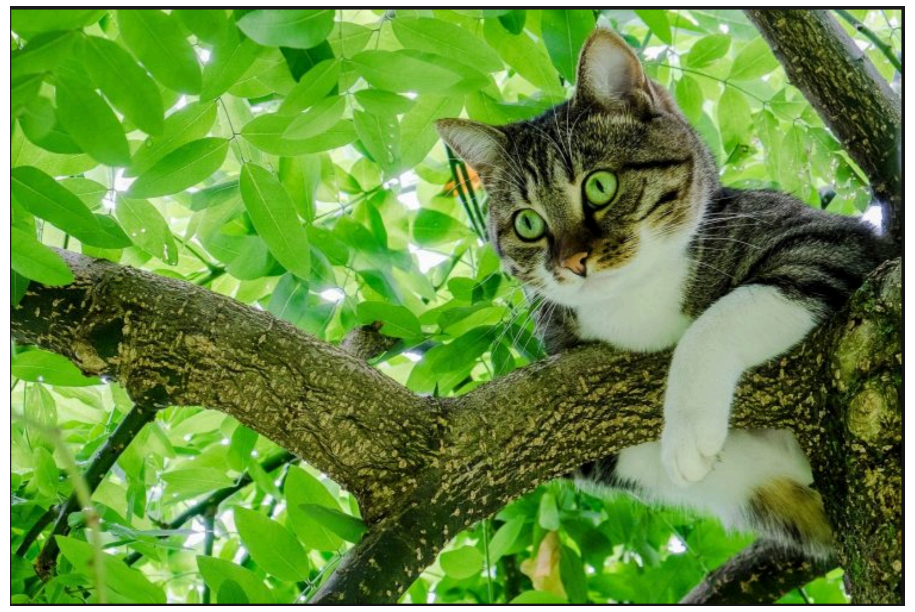
Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

Principles for Designing Datasets

"Neural Networks are Lazy"

training data



"cat"

trained network classification

"cat"

neural networks will always find the easiest way to solve a problem

(e.g., green background means "cat")

contributions from Sourya Dey

Principles for Designing Datasets

"Neural Networks are Lazy"

is ***the*** principle that should guide your dataset design

You want to maximize the ***coverage*** in your dataset

e.g., cats with non-green backgrounds were not covered in previous example

- Include **maximum diversity** in your dataset
  - **Think lazy** like a neural network and design your dataset for maximum coverage
  - Include difficult and **extremely difficult examples** in your dataset (even if you have to create them!)
    - Giving tough examples will not make your trained network worse at the easy cases!
  - You can never have too much (valid) data

How Much Data is Needed (MLPs)?

Sufficient Training-Sample Size for a Valid Generalization

Generalization is influenced by three factors: (1) the size of the training sample and how representative the training sample is of the environment of interest, (2) the architecture of the neural network, and (3) the physical complexity of the problem at hand. Clearly, we have no control over the lattermost factor. In the context of the other two factors, we may view the issue of generalization from two different perspectives:

- The architecture of the network is fixed (hopefully in accordance with the physical complexity of the underlying problem), and the issue to be resolved is that of determining the size of the training sample needed for a good generalization to occur.
- The size of the training sample is fixed, and the issue of interest is that of determining the best architecture of network for achieving good generalization.

Both of these viewpoints are valid in their own individual ways.

In practice, it seems that all we really need for a good generalization is to have the size of the training sample, N, satisfy the condition

$$\boxed{N = O\left(\frac{W}{\varepsilon}\right)} \tag{4.87}$$

where W is the total number of free parameters (i.e., synaptic weights and biases) in the network, ε denotes the fraction of classification errors permitted on test data (as in pattern classification), and $O(\cdot)$ denotes the order of quantity enclosed within. For example, with an error of 10 percent, the number of training examples needed should be about 10 times the number of free parameters in the network.

Equation (4.87) is in accordance with *Widrow's rule of thumb* for the LMS algorithm, which states that the settling time for adaptation in linear adaptive temporal filtering is approximately equal to the memory span of an adaptive tapped-delay-line filter divided by the misadjustment (Widrow and Stearns, 1985; Haykin, 2002). The misadjustment in the LMS algorithm plays a role somewhat analogous to the error ε in Eq. (4.87). Further justification for this empirical rule is presented in the next section.

(Number of parameters) divided by (target error rate)

Fashion MNIST Example

error rate ~ 0.1

training examples ~ 60,000

Suggests ~ 600K trainable parameters!

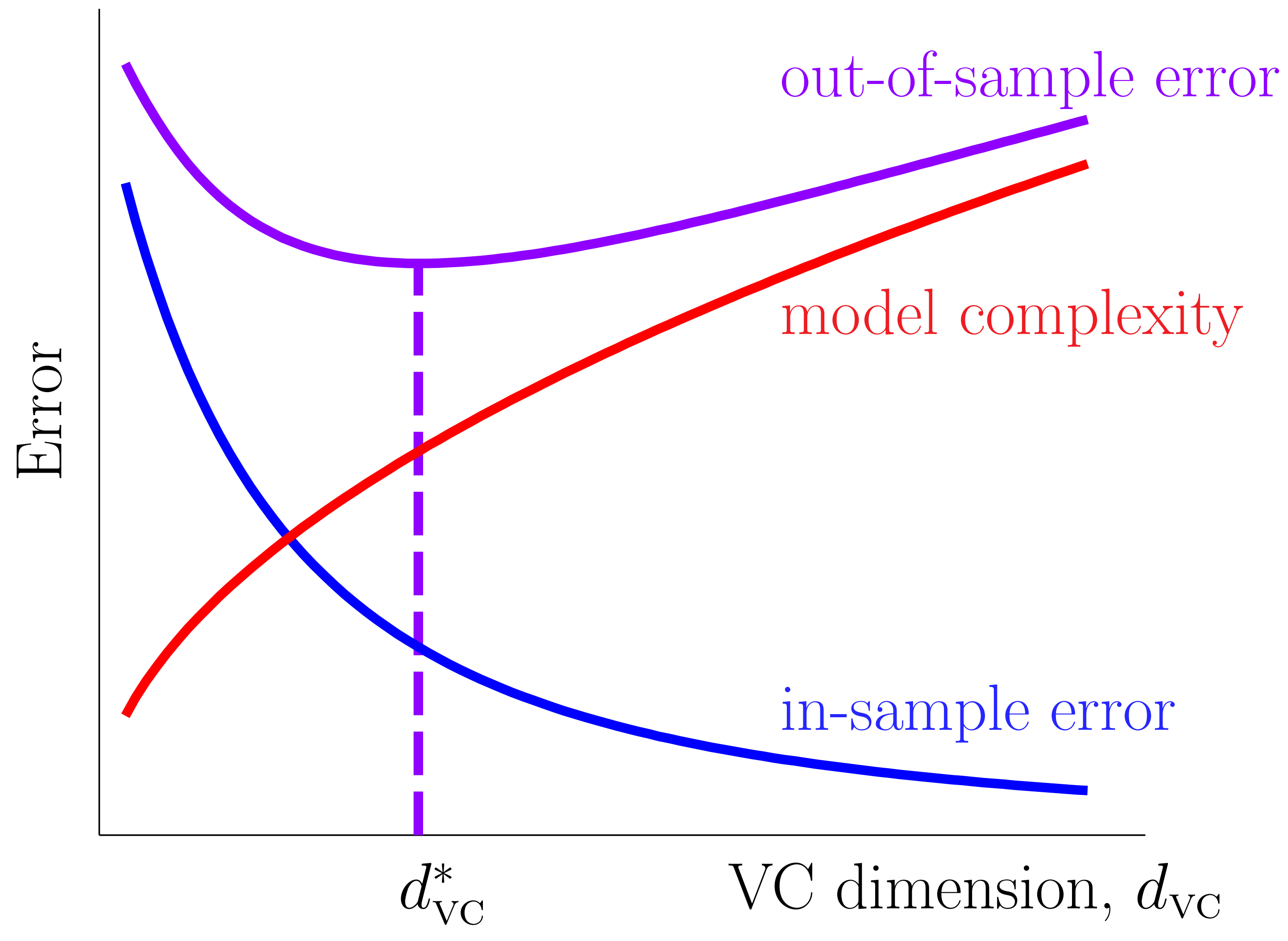
Obviously, this is a big-O rule of thumb!

[Haykin] Simon Haykin, Neural Networks And Learning Machines 3rd Edition, Pearson, 2009.

How Much Data is Needed (general)?

Vapnik-Chervonenkis (VC) Dimension

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + O\left(\sqrt{\frac{d_{\text{VC}} \log N}{N}}\right)$$



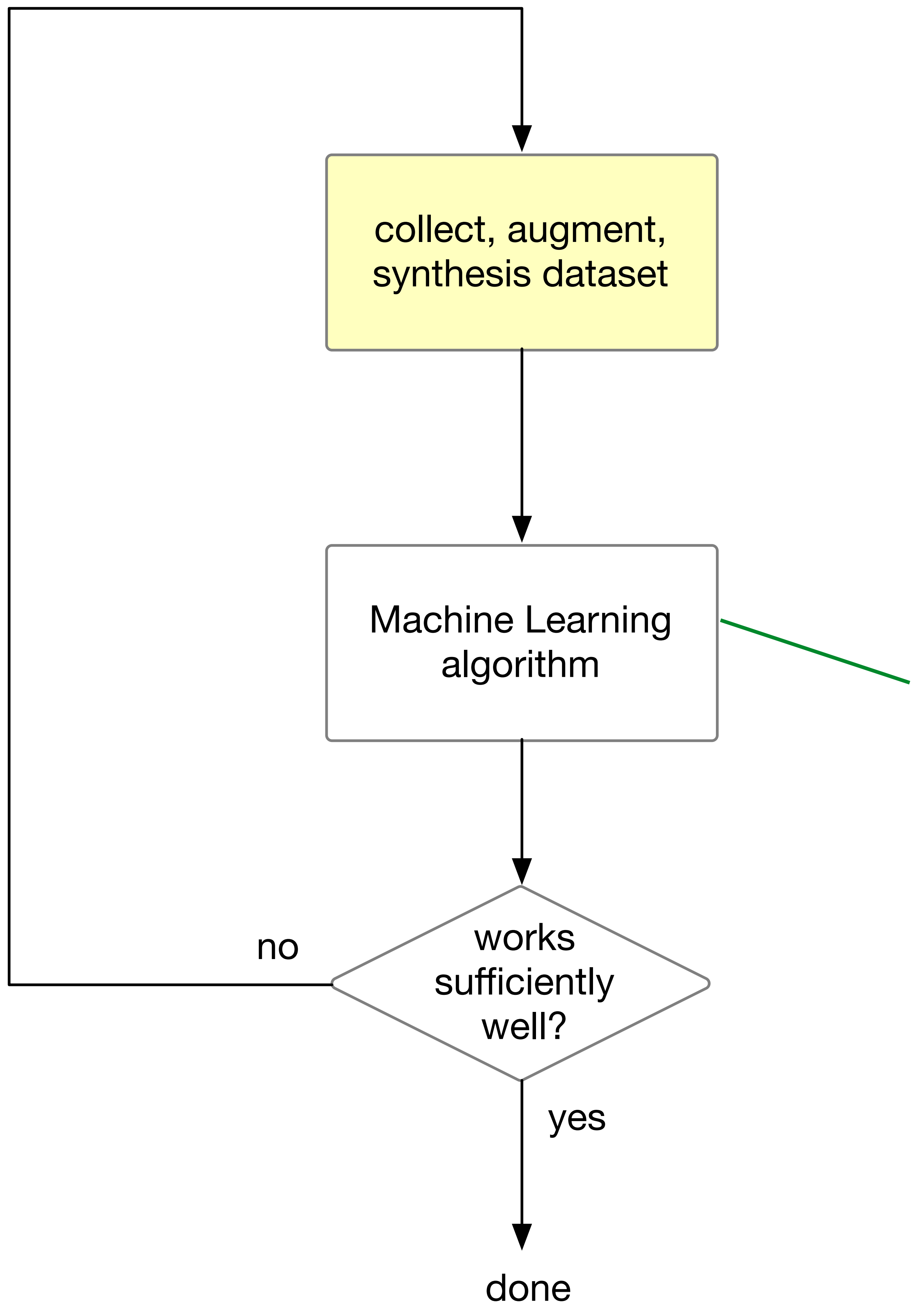
VC Dimension difficult to compute for complex (deep learning models)

Practical Rule of Thumb: $N = 10 \times d_{\mathbf{vc}}$

http://www.cs.rpi.edu/~magdon/courses/LFD-Slides/SlidesLect07.pdf

[AML] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Yien Lin, Learning from Data, A Short Course, AMLbook.com.

Principles for Designing Datasets



data (and ML) evaluated via end-to-end performance

much of the attention is here, but in practice, more iteration/time spent on data engineering

"all datasets are incomplete, but some have enough coverage to be useful"

Principles for Designing Datasets

Collection	make your neural network work (not be lazy) by giving examples of every scenario you expect it to work in
Labeling	labor intensive task (ways around this: synthetic data, use ML to label, $$$)
Coverage	make your neural network work (not be lazy) by giving examples of every scenario you expect it to work in
Contamination	accept that there will be mislabeled data in huge datasets due to human error or ambiguities
Cleaning	correct contamination effects, remove misleading or ambiguous examples. Automate this.
Augmentation	use synthetic means to produce better coverage and more difficult examples from your baseline data. In some cases, you may create synthetic data to augment your data

Principles for Designing Datasets

In practice, designing your dataset is the most important aspect in developing a deep-learning solution

90% of effort is spent on dataset design and maintenance (my experience)

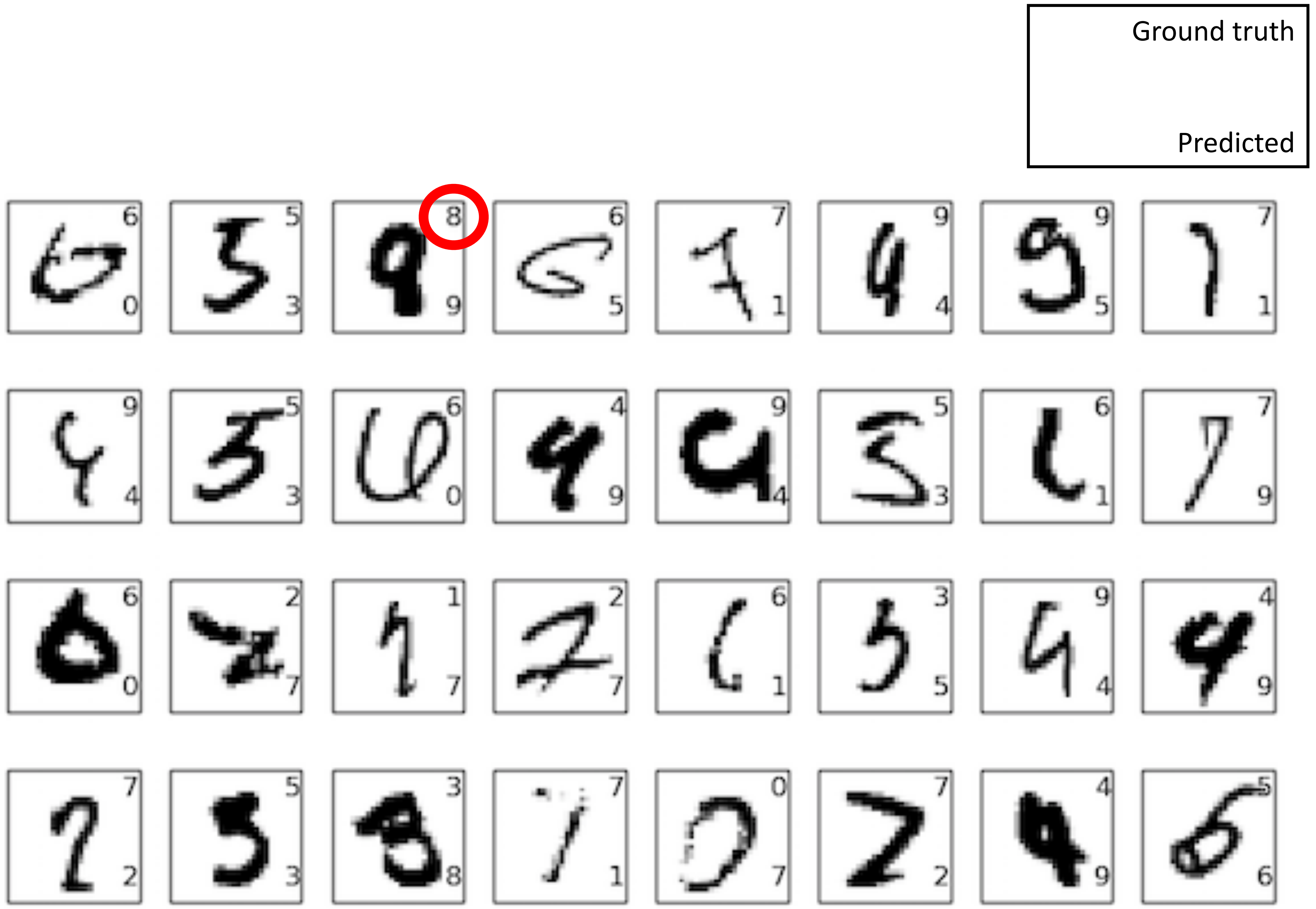
this is not apparent from reading papers and books because most materials focus on using publicly available datasets that serve as test benches

in our class, we are crowd-sourcing two datasets to try to illustrate the practical issues, but still not at a practical scale

Dataset Contamination / Cleaning

many datasets contain mislabeled or ambiguous data

“cat?” “dog?”

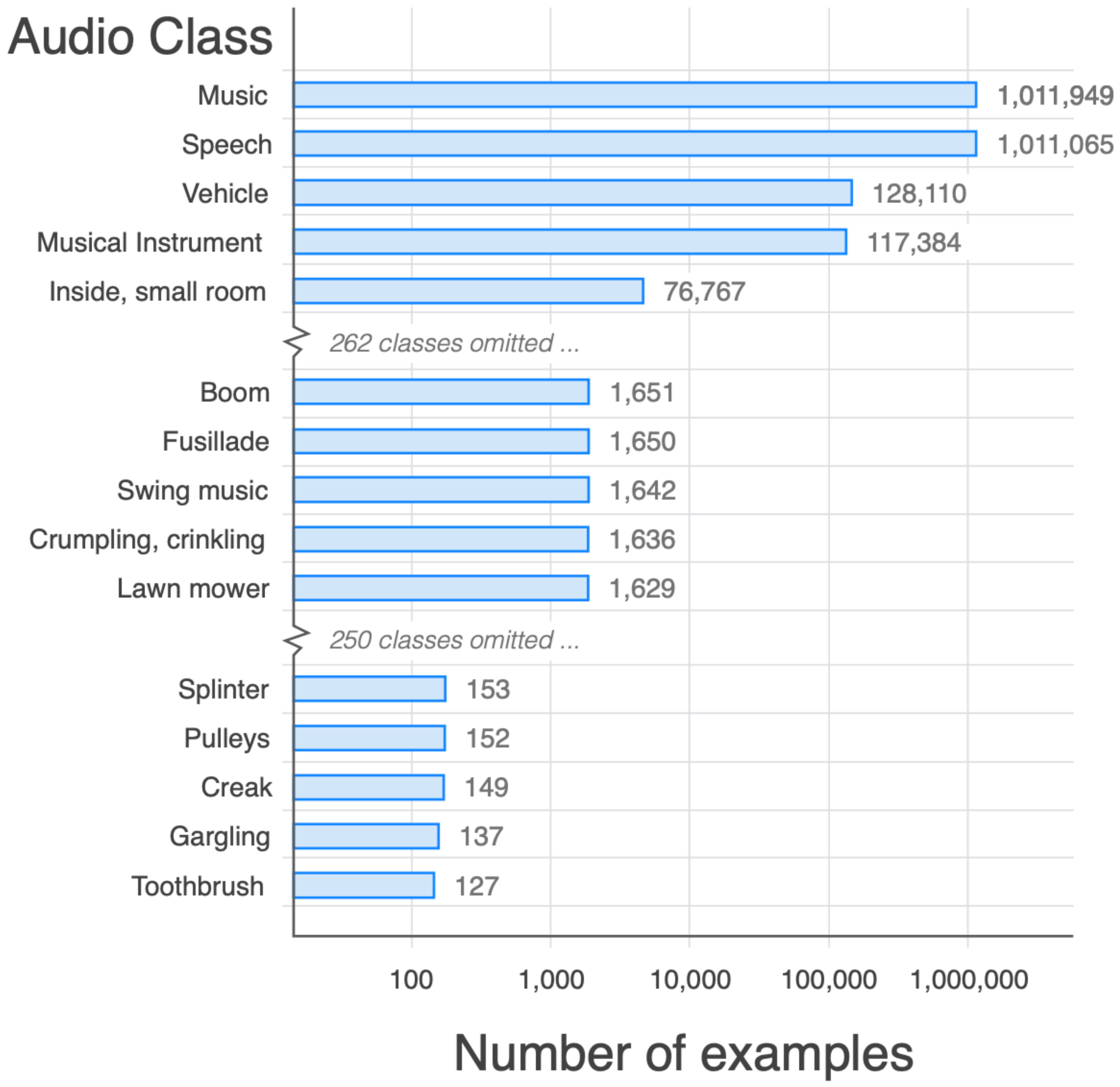


contributions from Sourya Dey

Dataset Contamination / Cleaning

Contamination may be relative to the inference task

example: Google's audioset



sample of "engine":

https://youtu.be/D9J91Iq52Bk?t=29

also has people speaking…

is this contamination?

task dependent

ex1: classify speech vs engines

ex2: classify jet engines vs car engines

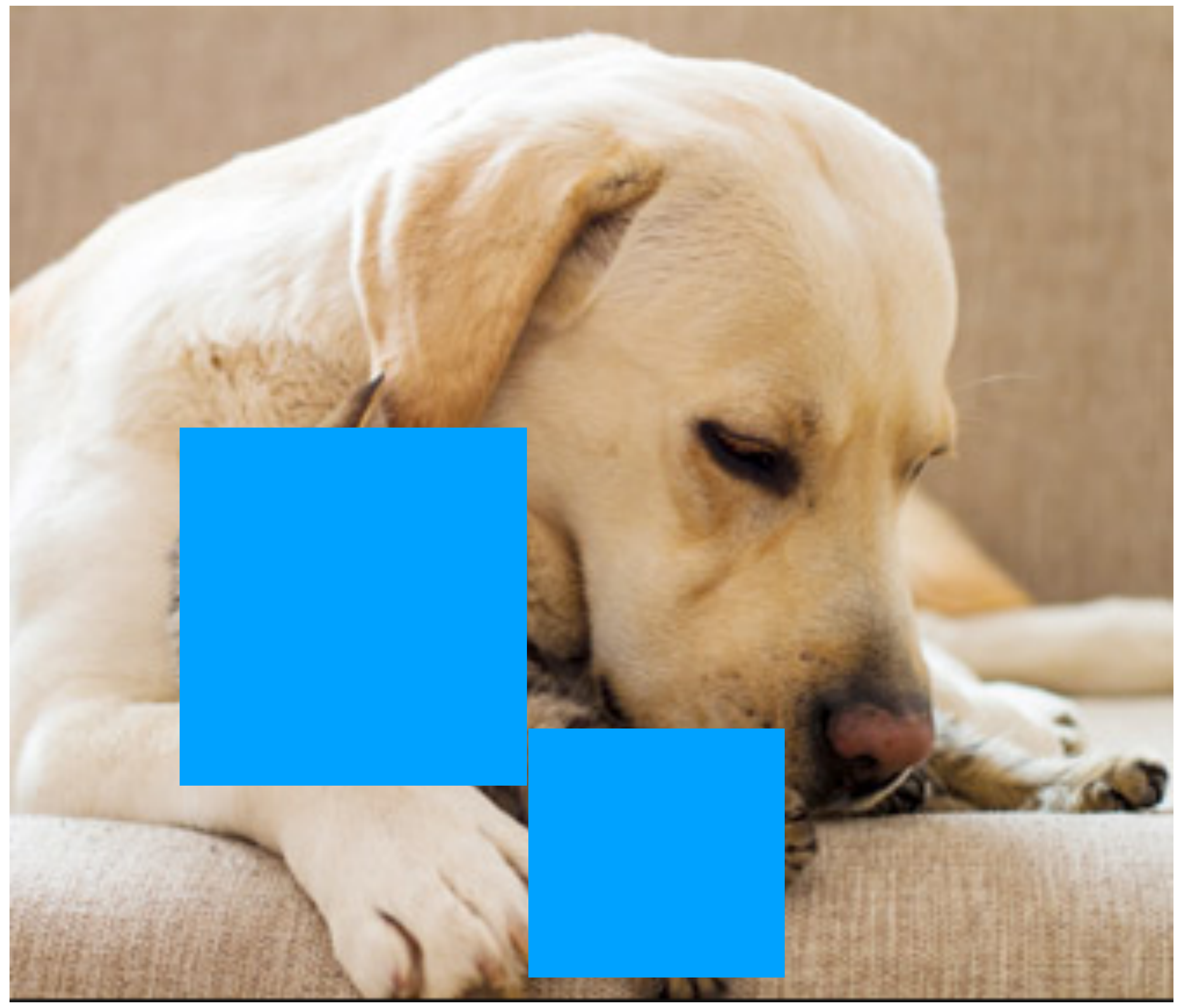
Dataset Contamination / Cleaning

Not unusual to have 5-10% contamination in your dataset

training will still work

if you expect ~99.9% accuracy on your task, contamination is more important than if you expect ~ 70% accuracy on your task

Dataset Contamination / Cleaning



Correct labels that are incorrect

Remove ambiguous examples

Mask/modify to make ambiguous examples less ambiguous

What could go wrong with this masking method in this example?

Dataset Contamination / Cleaning

Example: Adult Dataset
https://archive.ics.uci.edu/ml/datasets/adult

Features:
- Age
- Working class
- Education
- Marital status
- Occupation
- Race
- Sex
- Capital gain
- Hours per week
- Native country

how to handle missing fields in datasets?

Delete an entry — does this create a bias?

e.g., do low education responses leave blank fields?

Fill-in for missing data

replace numerical data by mean, e.g., age = 40

replace categorical data by mode, e.g., education = high school

replace missing data with a marker that can be incorporated into your loss —
e.g., age = -1 and write custom loss to not account for this age

Dataset Contamination / Cleaning

arXiv.org > cs > arXiv:3141.59265

Search... | All fields | Search

Help | Advanced Search

Computer Science > Machine Learning

Surpassing the state of the art on ImageNet by collecting more labels

(Submitted on 2020)

We achieve state-of-the-art 99.5% top-1 accuracy on ImageNet using a ResNet-50. This was achieved by paying people money to clean and grow the training set. First we cleaned up the incorrect labels in the existing training set and validation set. Then we found more unlabeled images similar to the high loss images in the validation set, labeled them, and added them to the training set. We repeated this process until accuracy improved enough. Data for this paper will be made available.

Subjects: **Machine Learning (cs.LG)**; Computer Vision and Pattern Recognition (cs.CV); Machine Learning (stat.ML)

Download:

- PDF
- Other formats

(license)

Current browse context:

cs.LG

< prev | next >

new | recent | 2002

Change to browse by:

cs

this is a joke from twitter, but makes the point

(Imagenet is one of the largest image classification datasets and often used as a benchmark)

Augmentation

increase the diversity and/or difficulty of your training data
through pre-processing

Examples (images):

rotate | flip | reflect | blur (change resolution)

add noise | "cut-out" (remove patches) | translate | camera modeling

Will see in CNNs that there are some nice built in image augmentation tools in python

Examples (audio):

add noise | add reverb | filter/ equalize | resample (change sample rates) | mic/speaker modeling

Example: English vs. Hindi vs. Mandarin

HW4: Computer Vision (CNN) problem (Jaili designing)

HW5: RNN for language classification

part of HW4 will be for you to generate audio samples for the language classification problem

what to do about….

silence?

noise?

mic (sample rate)?

speaker gender, age, accent?

without the class to generate, where would you get your data?

Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

Typical Flow for Deep Learning Development

General Good Practice

Get a good **baseline**:

solve the problem without a neural network first if possible

use a published baseline network as a baseline if you know the problem requires deep learning

Develop a **Full Dev Pipeline (scripts)**:

viewing/interpreting datasets and examples

cultivating/updating your dataset

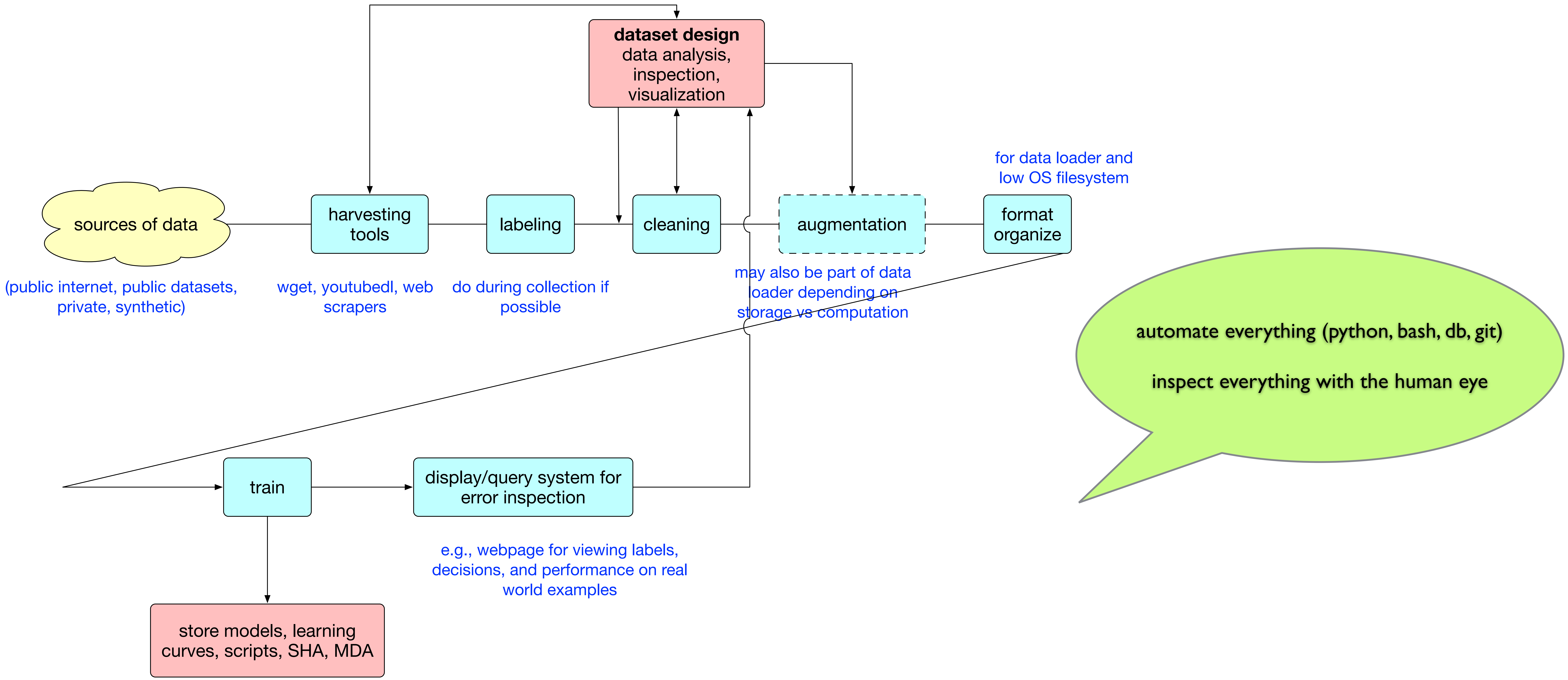
training with version control and auto-documentation

testing and visualization of the result on real-world data

keep your training simple to start

overfit a subset of data to make sure all is working before going big

Typical Flow for Deep Learning Development



popular [blog by Telsa Sr. Director of AI] hits many of these same points

Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

Common Data Normalization Methods

example: rating football (soccer) players

Feature	Units	Range
Height	Meters	1.5 to 2
Weight	Kilograms	50 to 100
Shot speed	Kmph	120 to 180
Shot curve	Degrees	0 to 10
Age	Years	20 to 35
Minutes played	Minutes	5,000 to 20,000
Fake diving?	--	Yes / No

different features on different scales…

normalize the data

contributions from Sourya Dey

Common Data Normalization Methods

recall: data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0^{\mathrm{t}} \\ \mathbf{x}_1^{\mathrm{t}} \\ \vdots \\ \mathbf{x}_{N-1}^{\mathrm{t}} \end{bmatrix} \qquad \mathbf{X}^{\mathrm{t}} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{N-1} \end{bmatrix}$$

$$\hat{\mathbf{m}}_{\mathbf{x}} = \left\langle \mathbf{x} \right\rangle_{\mathcal{D}} = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n$$

$$\begin{aligned} \hat{\mathbf{R}}_{\mathbf{x}} &= \left\langle \mathbf{x}\mathbf{x}^{\mathrm{t}} \right\rangle_{\mathcal{D}} = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n \mathbf{x}_n^{\mathrm{t}} \\ &= \frac{1}{N} \mathbf{X}^{\mathrm{t}} \mathbf{X} \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{K}}_{\mathbf{x}} &= \hat{\mathbf{R}}_{\mathbf{x}} - \hat{\mathbf{m}}_{\mathbf{x}} \hat{\mathbf{m}}_{\mathbf{x}}^{\mathrm{t}} \\ &= \left\langle [\mathbf{x} - \hat{\mathbf{m}}_{\mathbf{x}}] [\mathbf{x} - \hat{\mathbf{m}}_{\mathbf{x}}]^{\mathrm{t}} \right\rangle_{\mathcal{D}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} [\mathbf{x}_n - \hat{\mathbf{m}}_{\mathbf{x}}] [\mathbf{x}_n - \hat{\mathbf{m}}_{\mathbf{x}}]^{\mathrm{t}} \end{aligned}$$

contributions from Sourya Dey

Common Data Normalization Methods

feature-wise standardization

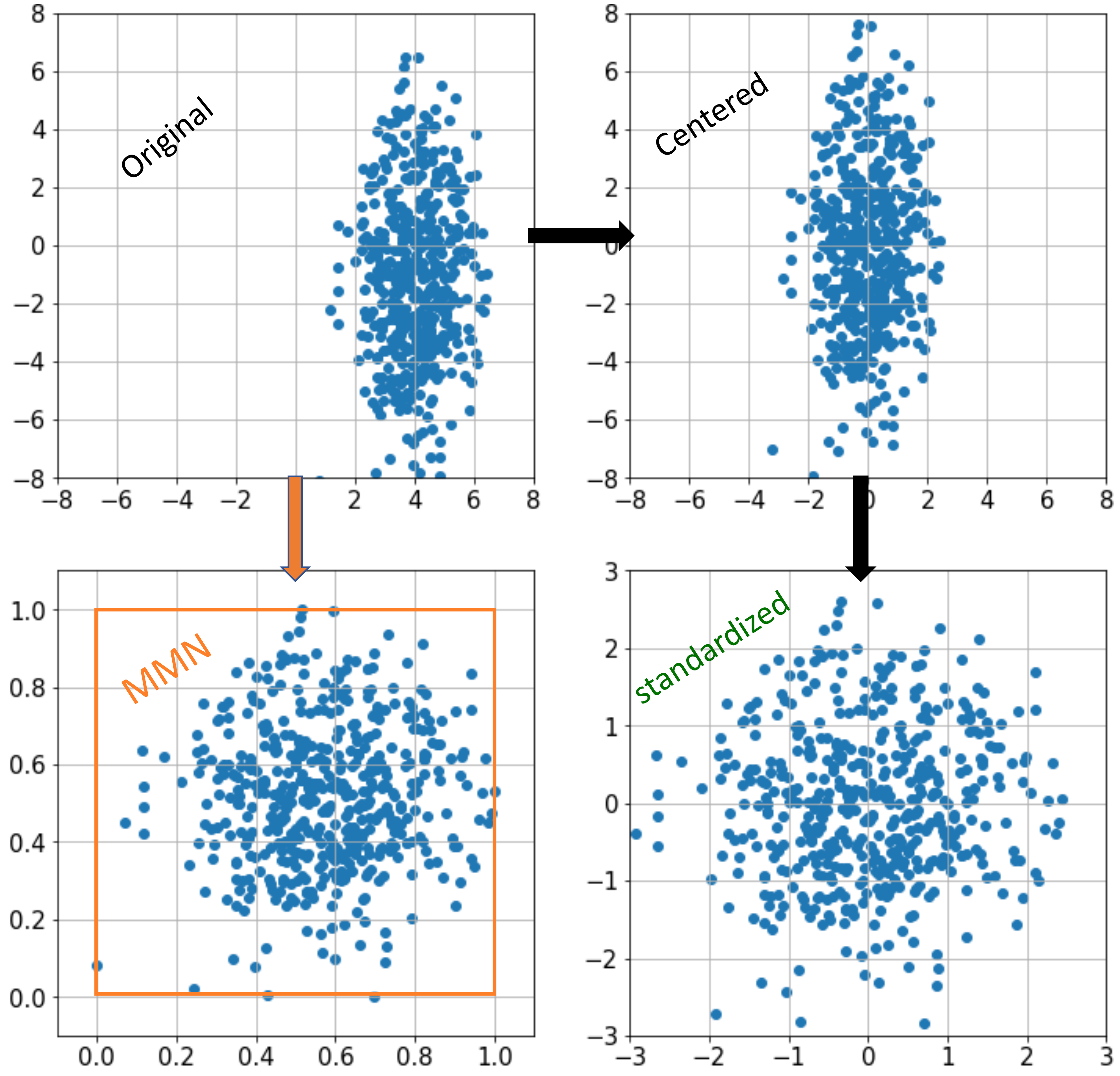
$$v_n[i] = \frac{x_n[i] - \hat{m}_x[i]}{\hat{\sigma}_x[i]}$$

scale each dimension of feature vector to mean 0, variance 1

feature-wise "minmax" normalization

$$v_n[i] = \frac{x_n[i] - \min_n x_n[i]}{\max_n x_n[i] - \min_n x_n[i]}$$

scale each dimension of feature vector to range [0,1]



contributions from Sourya Dey

Common Data Normalization Methods

previous methods do not account for feature correlation across dimensions…

do this by “whitening” the feature vector

yields a feature vector with uncorrelated, standard components

contributions from Sourya Dey

KL-Expansion

$$\mathbf{K_x}\mathbf{e}_k = \lambda_k \mathbf{e}_k \quad k = 0, 1, \ldots D-1 \qquad \text{(Eigen equation)}$$

$$\mathbf{e}_k^{\mathrm{t}}\mathbf{e}_l = \delta[k-l] \quad \lambda_k \geq 0 \qquad \text{(orthonormal e-vectors)}$$

$$\mathbf{x}(u) = \sum_{k=0}^{D-1} X_k(u)\mathbf{e}_k \qquad \text{(change of coordinates)}$$

$$X_k(u) = \mathbf{e}_k^{\mathrm{t}}\mathbf{x}(u)$$

$$\mathbb{E}\left\{X_k(u)X_l(u)\right\} = \mathbf{e}_k^{\mathrm{t}}\mathbf{K_x}\mathbf{e}_l = \lambda_k\delta[k-l] \qquad \text{(uncorrelated components)}$$

$$\mathbf{K_x} = \sum_{k=0}^{N-1} \lambda_k \mathbf{e}_k\mathbf{e}_k^{\mathrm{t}} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{\mathrm{t}} \qquad \text{(Mercer's Theorem)}$$

$$\mathbb{E}\left\{\|\mathbf{x}(u)\|^2\right\} = \mathrm{tr}\left(\mathbf{K_x}\right) = \sum_{k=0}^{D-1} \lambda_k \qquad \text{(Total Energy)}$$

Always exists because K is nnd-symmetric

KL-Expansion

Can always find orthonormal set of e-vectors of K

These are an alternate coordinate systems (rotations, reflections)

in this eigen-coordinate system, the components are uncorrelated

(principle components)

The eigen-values are the variant (energy) in each of these principle directions

(can be used to reduce dimensions by throwing out components with low energy)

KL-Expansion

$$d_k(u) = \mathbf{e}_k^{\mathrm{t}} \mathbf{x}(u) \qquad k = 0, 1, \ldots D-1$$

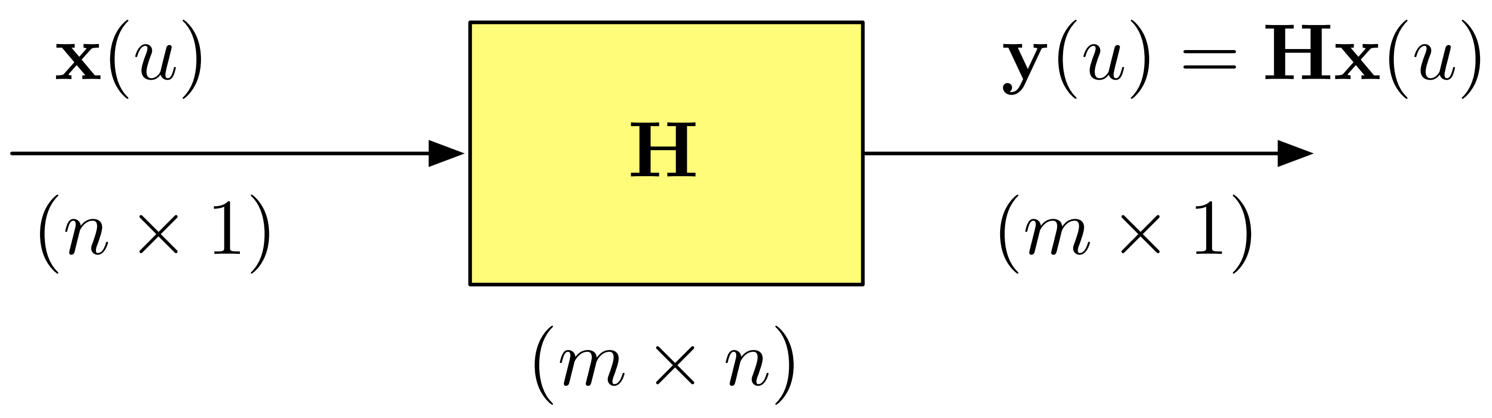
$$\mathbf{d}(u) = \mathbf{E}^{\mathrm{t}} \mathbf{x}(u)$$

$$\begin{aligned} \mathbf{K}_{\mathbf{d}} &= \mathbf{E}^{\mathrm{t}} \mathbf{K}_{\mathbf{x}} \mathbf{E} \\ &= \mathbf{E}^{\mathrm{t}} \left(\mathbf{E} \boldsymbol{\Lambda} \mathbf{E}^{\mathrm{t}} \right) \mathbf{E} \\ &= \boldsymbol{\Lambda} = \mathbf{diag}(\lambda_k) \end{aligned}$$

Multiplying by E^t makes the components uncorrelated

$$\mathbf{E} = \left[\begin{array}{c|c|c|c|c} \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_{D-1} \end{array} \right]$$

Random Vectors



$$\mathbf{m}_{\mathbf{y}} = \mathbf{H}\mathbf{m}_{\mathbf{x}}$$

$$\mathbf{R}_{\mathbf{y}} = \mathbf{H}\mathbf{R}_{\mathbf{x}}\mathbf{H}^{\mathrm{t}}$$

$$\mathbf{K}_{\mathbf{y}} = \mathbf{H}\mathbf{K}_{\mathbf{x}}\mathbf{H}^{\mathrm{t}}$$

Special case

$$y(u) = \mathbf{b}^{\mathrm{t}}\mathbf{x}(u) \qquad (1 \times 1)$$

$$m_y = \mathbf{b}^{\mathrm{t}}\mathbf{m}_{\mathbf{x}}$$

$$\mathbb{E}\left\{y^2(u)\right\} = \mathbf{b}^{\mathrm{t}}\mathbf{R}_{\mathbf{x}}\mathbf{b}$$

$$\sigma_y^2 = \mathbf{b}^{\mathrm{t}}\mathbf{K}_{\mathbf{x}}\mathbf{b}$$

example math

$$\begin{aligned}
\mathbf{R}_{\mathbf{y}} &= \mathbb{E}\left\{\mathbf{y}(u)\mathbf{y}^{\mathrm{t}}(u)\right\} \\
&= \mathbb{E}\left\{(\mathbf{H}\mathbf{x}(u))(\mathbf{H}\mathbf{x}(u))^{\mathrm{t}}\right\} \\
&= \mathbb{E}\left\{\mathbf{H}\mathbf{x}(u)\mathbf{x}^{\mathrm{t}}(u)\mathbf{H}^{\mathrm{t}}\right\} \\
&= \mathbf{H}\mathbb{E}\left\{\mathbf{x}(u)\mathbf{x}^{\mathrm{t}}(u)\right\}\mathbf{H}^{\mathrm{t}} \\
&= \mathbf{H}\mathbf{R}_{\mathbf{x}}\mathbf{H}^{\mathrm{t}}
\end{aligned}$$

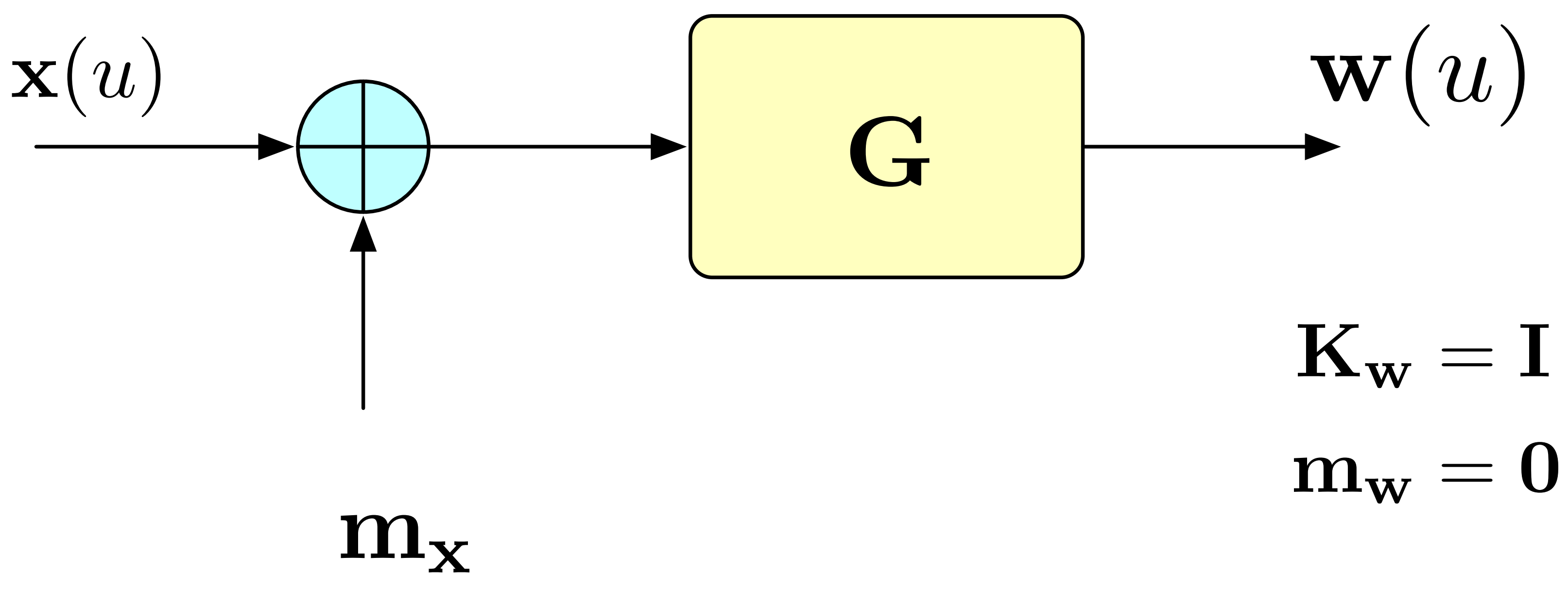
Note that covariance/correlation matrices are symmetric, non-negative definite

KL-Expansion - Relation to Whitening

$$w_k(u) = \frac{X_k(u)}{\sqrt{\lambda_k}} = \frac{\mathbf{e}_k^{\mathrm{t}}\mathbf{x}(u)}{\sqrt{\lambda_k}} \qquad k = 0, 1, \ldots D-1$$

$$\mathbf{w}(u) = \mathbf{\Lambda}^{-1/2}\mathbf{E}^{\mathrm{t}}\mathbf{x}(u)$$

$$\begin{aligned}\mathbf{K_w} &= \mathbf{\Lambda}^{-1/2}\mathbf{E}^{\mathrm{t}}\mathbf{K_x}\mathbf{E}\mathbf{\Lambda}^{-1/2} \\ &= \mathbf{\Lambda}^{-1/2}\mathbf{\Lambda}\mathbf{\Lambda}^{-1/2} \\ &= \mathbf{I}\end{aligned}$$



$$\mathbf{K_w} = \mathbf{I}$$
$$\mathbf{m_w} = \mathbf{0}$$

For any orthogonal matrix U, this whitening matrix also works:

$$\mathbf{G} = \mathbf{U}\mathbf{\Lambda}^{-1/2}\mathbf{E}^{\mathrm{t}}$$

$$\mathbf{K_x} = \mathbf{H}\mathbf{H}^{\mathrm{t}} \implies \mathbf{G} = \mathbf{H}^{-1}$$

Data Normalization - Whitening

Do this using the sample statistics over the training data

normalizes each feature vector component to mean 0, variance 1

whitened feature components are uncorrelated

all feature vector components are equally important

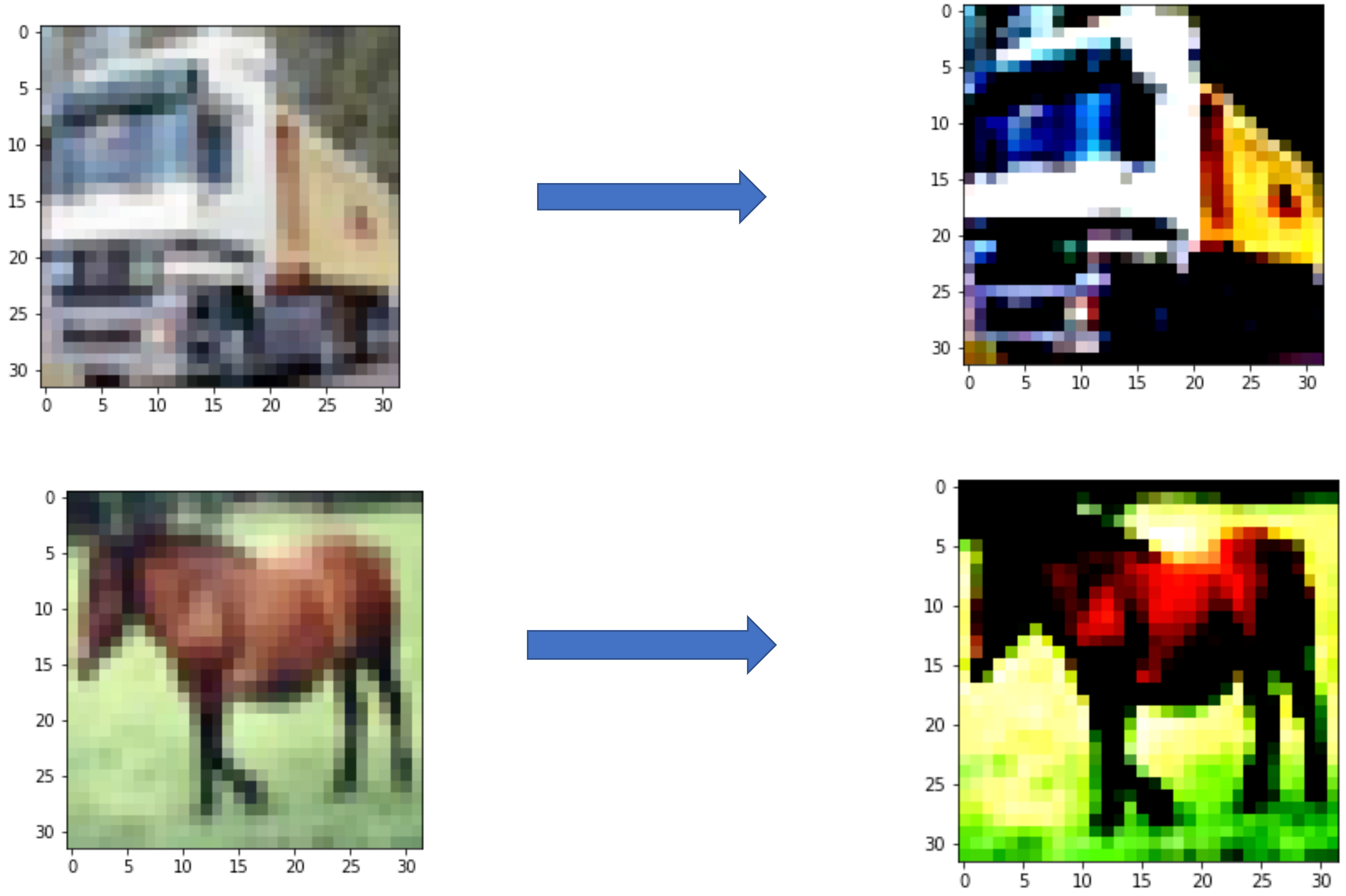
aka: “zero component analysis”

contributions from Sourya Dey

Data Normalization - Global Contrast Normalization

$\boldsymbol{x}$: Single image

$$x_{\text{pixel}} = \frac{x_{\text{pixel}} - \mu_{(\text{all pixels in } \boldsymbol{x})}}{\sigma_{(\text{all pixels in } \boldsymbol{x})}}$$



Increase contrast (standard deviation of pixels) of each image, one at a time

contributions from Sourya Dey

Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

KL-Expansion - Relation to PCA

$$\tilde{x}_k(u) = \mathbf{e}_k^{\mathrm{t}} \mathbf{x}(u) \qquad k = 0, 1, \ldots T-1$$

$$\tilde{\mathbf{x}}(u) = \mathbf{E}_{[:T]}^{\mathrm{t}} \mathbf{x}(u) \qquad \text{first } T \text{ components}$$

$$\mathbf{K}_{\tilde{\mathbf{x}}} = \mathbf{\Lambda}_{[:T]} \qquad \text{assumes ordered e-values: } \lambda_0 \geq \lambda_1 \geq \ldots \lambda_{D-1}$$

$$\mathbb{E}\left\{\|\tilde{\mathbf{x}}(u)\|^2\right\} = \sum_{k=0}^{T-1} \lambda_k$$

$$\mathbb{E}\left\{\|\mathbf{x}(u) - \tilde{\mathbf{x}}(u)\|^2\right\} = \sum_{k=T}^{D-1} \lambda_k \qquad \text{minimizes approximation error (lossy compression)}$$

PCA is simply taking only the T most important e-directions or principle components

$$\mathbf{E}_{[:T]} = \left[\begin{array}{c|c|c|c|c} \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_{T-1} \end{array} \right]$$

KL/PCA for Data

Everything is the same, except we use data-averaging instead of E{.}

$$\hat{\mathbf{R}}_{\mathbf{x}} = \left\langle \mathbf{x}\mathbf{x}^{\mathrm{t}} \right\rangle_{\mathcal{D}}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n \mathbf{x}_n^{\mathrm{t}}$$

$$= \frac{1}{N} \mathbf{X}^{\mathrm{t}} \mathbf{X}$$

Both KL/PCA can be applied to R or K. Center x if you want to use K

x <— x - m

(same if mean is zero)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0^{\mathrm{t}} \\ \mathbf{x}_1^{\mathrm{t}} \\ \vdots \\ \mathbf{x}_{N-1}^{\mathrm{t}} \end{bmatrix} \qquad \mathbf{X}^{\mathrm{t}} = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{N-1} \end{bmatrix}$$

“stacked” data matrix

KL/PCA for Data

PCA for data

$$\tilde{\mathbf{x}}_n = \mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_n \qquad \text{first } T \text{ components}$$

$$\mathbf{E}_{[:T]} = \left[\begin{array}{c|c|c|c|c} \mathbf{e}_0 & \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_{T-1} \end{array} \right]$$

apply to the "stacked" data matrix

$$\tilde{\mathbf{X}} = \begin{bmatrix} \left(\mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_0\right)^{\mathrm{t}} \\ \left(\mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_1\right)^{\mathrm{t}} \\ \vdots \\ \left(\mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_{N-1}\right)^{\mathrm{t}} \end{bmatrix} = \mathbf{X}\mathbf{E}_{[:T]}$$

$$\tilde{\mathbf{X}}^{\mathrm{t}} = \left[\begin{array}{cccc} \mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_0 & \mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_1 & \cdots & \mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{x}_{N-1} \end{array} \right] = \mathbf{E}^{\mathrm{t}}_{[:T]} \mathbf{X}^{\mathrm{t}}$$

$$\underset{N\times T}{\tilde{\mathbf{X}}} = \underset{N\times D}{\mathbf{X}} \quad \underset{D\times T}{\mathbf{E}_{[:T]}}$$

$$\underset{T\times T}{\tilde{\mathbf{X}}^{\mathrm{t}}\tilde{\mathbf{X}}}$$

dimension reduced from D to T

KL/PCA for Data — relation to SVD

SVD for an arbitrary matrix A

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times m}{\mathbf{U}} \quad \underset{m\times n}{\mathbf{\Sigma}} \quad \underset{n\times n}{\mathbf{V}^{\mathrm{t}}}$$

U, V are orthogonal matrices, sigma is "diagonal" with singular values on diagonal

Use SVD to expand matrix A^t A

$$\mathbf{A}^{\mathrm{t}}\mathbf{A} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V})^{\mathrm{t}}\,\mathbf{U}\mathbf{\Sigma}\mathbf{V}$$

$$= \mathbf{V}\mathbf{\Sigma}^{\mathrm{t}}\mathbf{U}^{\mathrm{t}}\mathbf{U}\mathbf{\Sigma}\mathbf{V}$$

$$= \underset{n\times n}{\mathbf{V}} \quad \underset{n\times n}{\mathbf{\Sigma}\mathbf{\Sigma}^{\mathrm{t}}} \quad \underset{n\times n}{\mathbf{V}^{\mathrm{t}}}$$

$$= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{\mathrm{t}}$$

The SVD for A provides the KL factorization for the non-negative definition, symmetric matrix A^t A

Note that this is also the SVD for A^t A

KL/PCA for Data — relation to SVD

SVD for stacked data matrix X

$$\underset{N\times D}{\mathbf{X}} = \underset{N\times N}{\mathbf{U}} \quad \underset{N\times D}{\mathbf{\Sigma}} \quad \underset{D\times D}{\mathbf{V}^{\mathrm{t}}}$$

$$\underset{D\times D}{\mathbf{X}^{\mathrm{t}}\mathbf{X}} = \underset{D\times D}{\mathbf{V}} \quad \underset{D\times D}{\mathbf{\Sigma}\mathbf{\Sigma}^{\mathrm{t}}} \quad \underset{D\times D}{\mathbf{V}^{\mathrm{t}}}$$

$$= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^{\mathrm{t}}$$

Equivalent approaches:

1) Find SVD of X, take V

2) Find Eigen decomposition of X^t X, take E = V

3) Find SVD of X^t X, take V = U = E

$$\underset{N\times T}{\tilde{\mathbf{X}}} = \underset{N\times D}{\mathbf{X}} \quad \underset{D\times T}{\mathbf{V}_{[:T]}}$$

KL/PCA for Data — relation to SVD

Equivalent approaches:

1) Find SVD of X, take V

2) Find Eigen decomposition of X^t X, take E = V

3) Find SVD of X^t X, take V = U = E

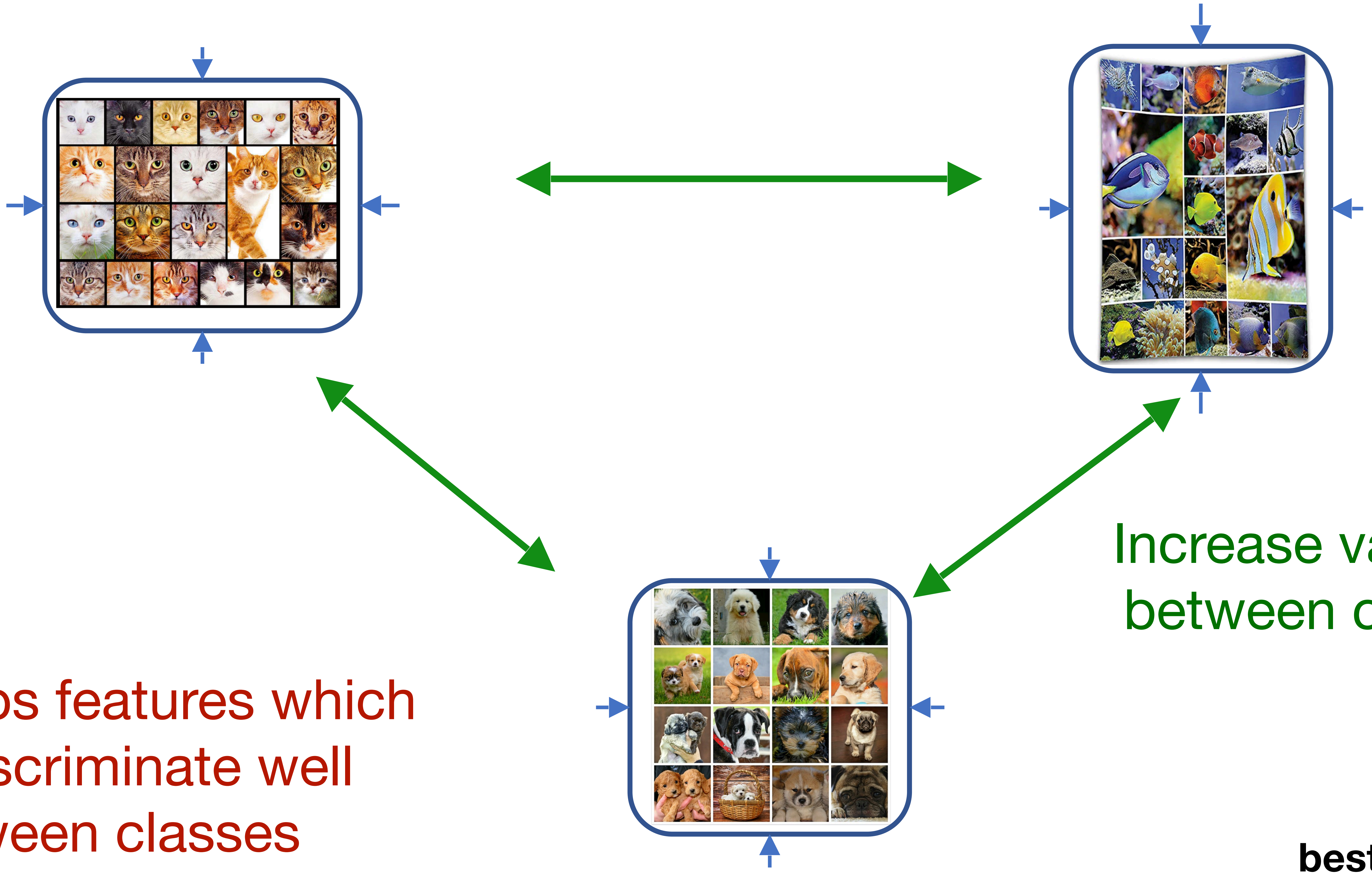
$$\underset{N\times T}{\tilde{\mathbf{X}}} = \underset{N\times D}{\mathbf{X}} \quad \underset{D\times T}{\mathbf{V}_{[:T]}}$$

Sourya noted that he uses method 3, with numpy.linalg.svd, instead of method 2, with numpy.linalg.eig, since the SVD returns the e-vectors in sorted order and Eig does not

contributions from Sourya Dey

Linear Discriminant Analysis (LDA)

Similar to PCA but is supervised and for classification problem



Decrease variance within a class

Increase variance between classes

LDA keeps features which can discriminate well between classes

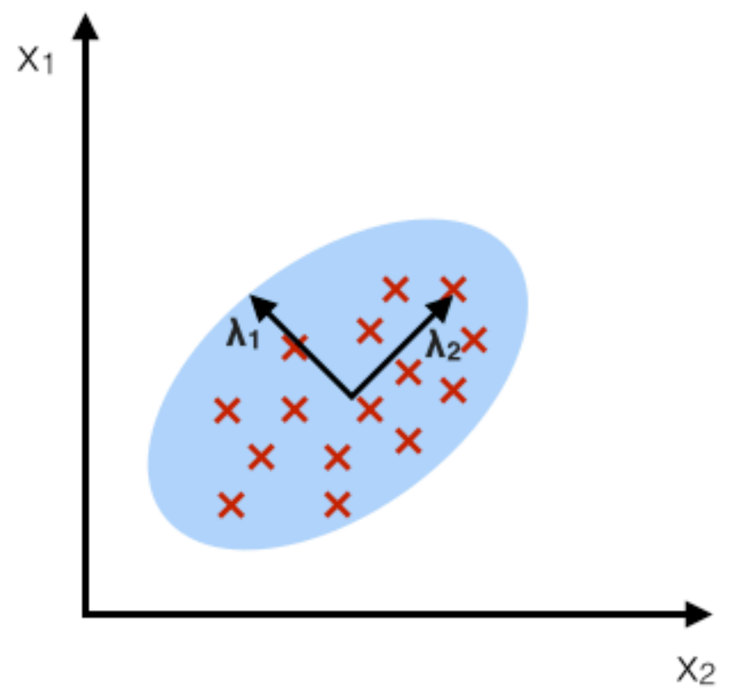
best reference is this blog

contributions from Sourya Dey

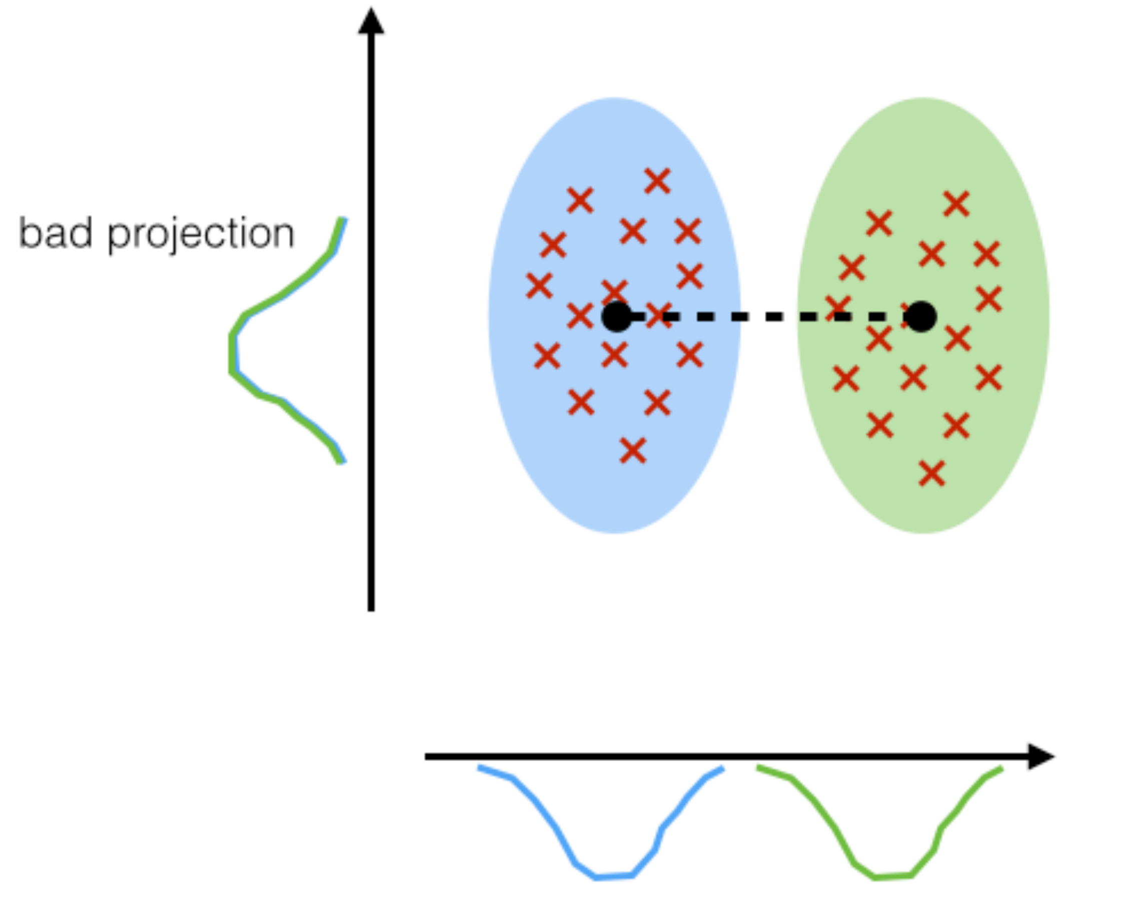
Linear Discriminant Analysis (LDA)

Similar to PCA but is supervised and for classification problem

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



best reference is this blog

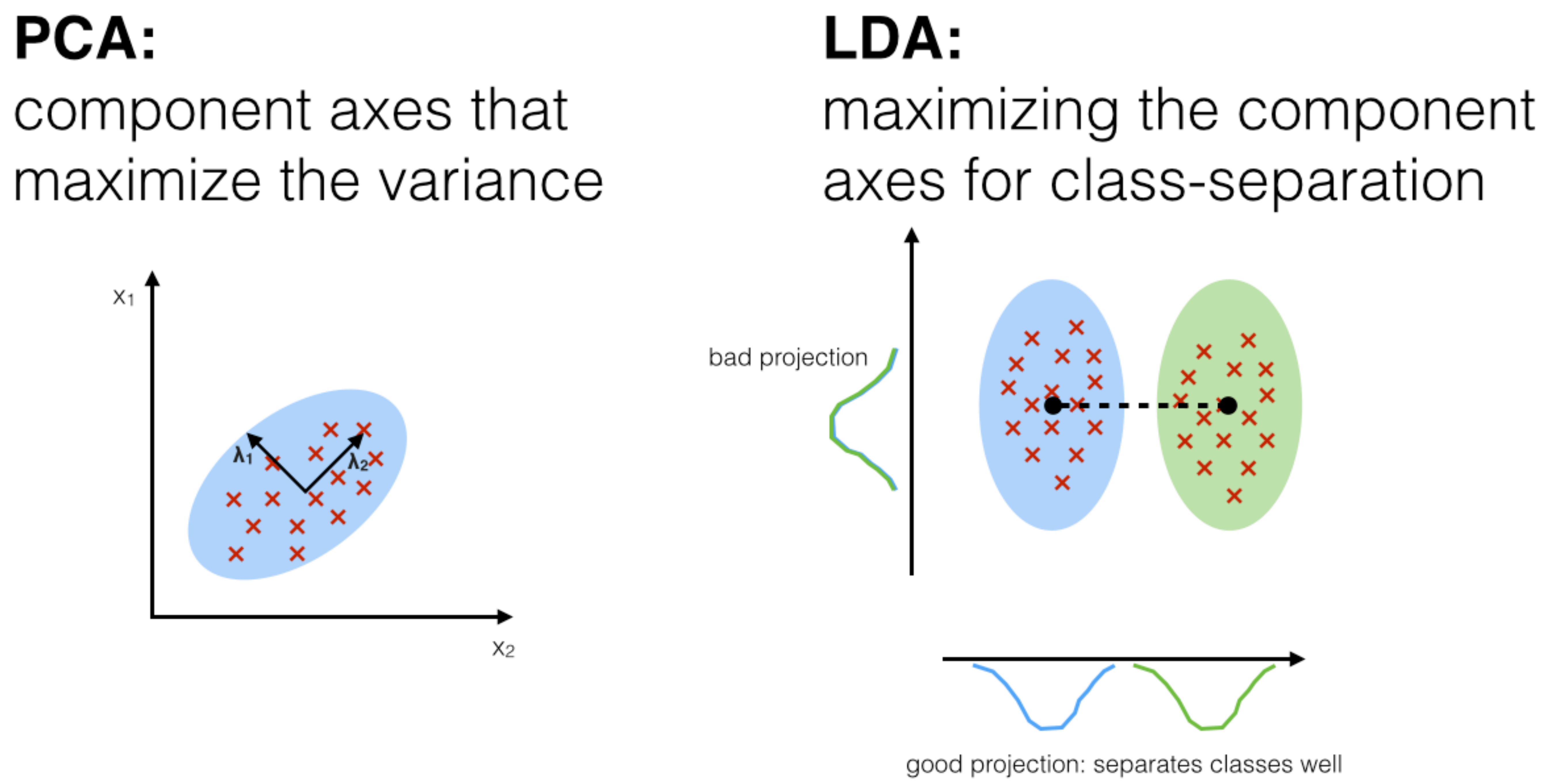
contributions from Sourya Dey

Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

Linear Discriminant Analysis (LDA)

Similar to PCA but is supervised and for classification problem



best reference is this blog

contributions from Sourya Dey

Pre-Processing (PCA, LDA, Normalization)

Use statistics collected from the **Training Data** only

apply same transformation to training, val, test data

Note that many of these techniques can be viewed as a fixed linear layer at the start of the network that is not trained as part of BP

alternative is to have a “bottleneck" layer for dimensionality reduction (i.e., learn ~ LDA during BP)

batch-normalization similarly is a method of learning normalization

Outline for Slides

- Principles for designing datasets
- Typical flow for deep learning development
- Common normalization methods
- PCA and LDA for dimensionality reduction
- Where to find data and how to grab it

Datasets Available in tf.keras

https://keras.io/datasets/

https://www.tensorflow.org/api_docs/python/tf/keras/datasets

`boston_housing` module: Boston housing price regression dataset.

`cifar10` module: CIFAR10 small images classification dataset.

`cifar100` module: CIFAR100 small images classification dataset.

`fashion_mnist` module: Fashion-MNIST dataset.

`imdb` module: IMDB sentiment classification dataset.

`mnist` module: MNIST handwritten digits dataset.

`reuters` module: Reuters topic classification dataset.

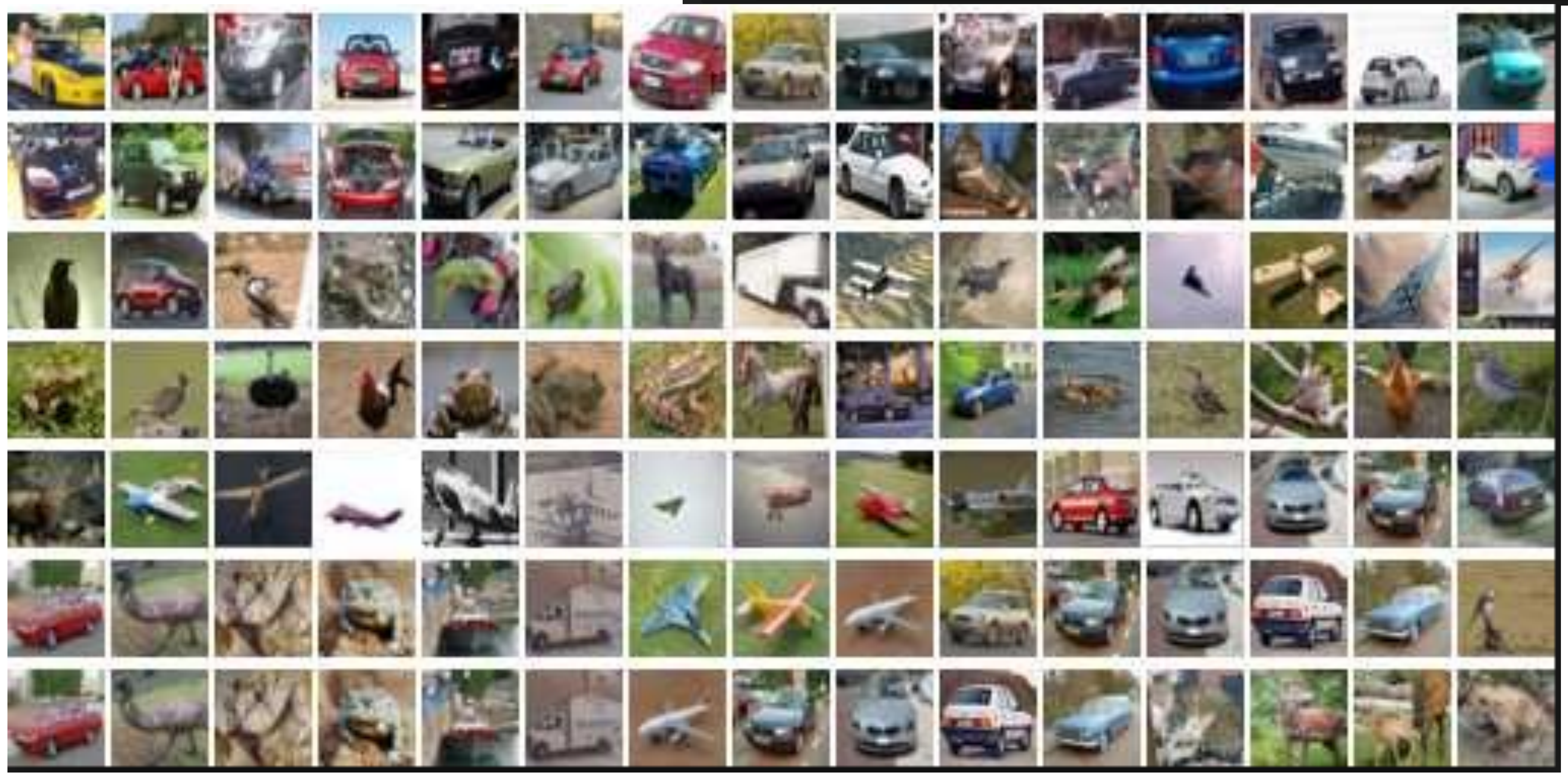
small number of “built-in” datasets to get started experimenting

Datasets Available in tf.keras

- **MNIST (MLP / CNN):**
  - 28x28 images, 10 classes
  - Initial benchmark
  - SOTA testacc: >99%

- **Fashion MNIST (MLP / CNN):**
  - 28x28 images, 10 classes
  - More challenging than MNIST, but same parameters
  - SOTA testacc: ~94%

- **CIFAR-10, -100 (CNN):**
  - 32x32x3 images (RGB), 10 or 100 classes
  - Widely used benchmark
  - SOTA testacc: ~97%, ~84%



Datasets Available in tf.keras

```python
from tensorflow.keras.datasets import imdb

(x_train, y_train), (x_test, y_test) = imdb.load_data(path="imdb.npz",
                                                      num_words=None,
                                                      skip_top=0,
                                                      maxlen=None,
                                                      seed=113,
                                                      start_char=1,
                                                      oov_char=2,
                                                      index_from=3)
```

- **IMDB Movie reviews sentiment classification**
  - 25,000 movies reviews from IMDB
  - labeled by sentiment (positive/negative)
  - words are indexed by overall frequency in the dataset

- **Reuters newswire topics classification**
  - 11,228 newswires from Reuters, labeled over 46 topics. As with the IMDB dataset, each wire is encoded as a sequence of word indexes (same conventions).

- **Boston housing price**
  - 13 attributes (features) of houses at different locations around the Boston suburbs in the late 1970s
  - labels are hous prices (late 1970s)

Datasets Available in Pytorch

image datasets

Datasets

- MNIST
- Fashion-MNIST
- KMNIST
- EMNIST
- QMNIST
- FakeData
- COCO
  - Captions
  - Detection
- LSUN
- ImageFolder
- DatasetFolder
- ImageNet
- CIFAR
- STL10
- SVHN
- PhotoTour
- SBU
- Flickr
- VOC
- Cityscapes
- SBD
- USPS
- Kinetics-400
- HMDB51
- UCF101

text datasets

Datasets

- Language Modeling
  - WikiText-2
  - WikiText103
  - PennTreebank
- Sentiment Analysis
  - SST
  - IMDb
- Text Classification
  - TextClassificationDataset
  - AG_NEWS
  - SogouNews
  - DBpedia
  - YelpReviewPolarity
  - YelpReviewFull
  - YahooAnswers
  - AmazonReviewPolarity
  - AmazonReviewFull
- Question Classification
  - TREC
- Entailment
  - SNLI
  - MultiNLI
- Language Modeling
  - WikiText-2
  - WikiText103
  - PennTreebank
- Machine Translation
  - Multi30k
  - IWSLT
  - WMT14
- Sequence Tagging
  - UDPOS
  - CoNLL2000Chunking
- Question Answering
  - BABI20
- Unsupervised Learning
  - EnWik9

audio datasets

Datasets

- COMMONVOICE
- LIBRISPEECH
- VCTK
- YESNO

"built-in" to tf.keras or pytorch just means that there is a loader and the framework will handle the initial download

Common Datasets in Computer Vision

- **Imagenet:**
  - > 14M images, 224x224x3, with 1000 classes
  - Common benchmark for image classification
  - SOTA testacc: >84%

tf.keras has many of the popular image classification networks already pre-defined and also pre-trained on imagenet

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

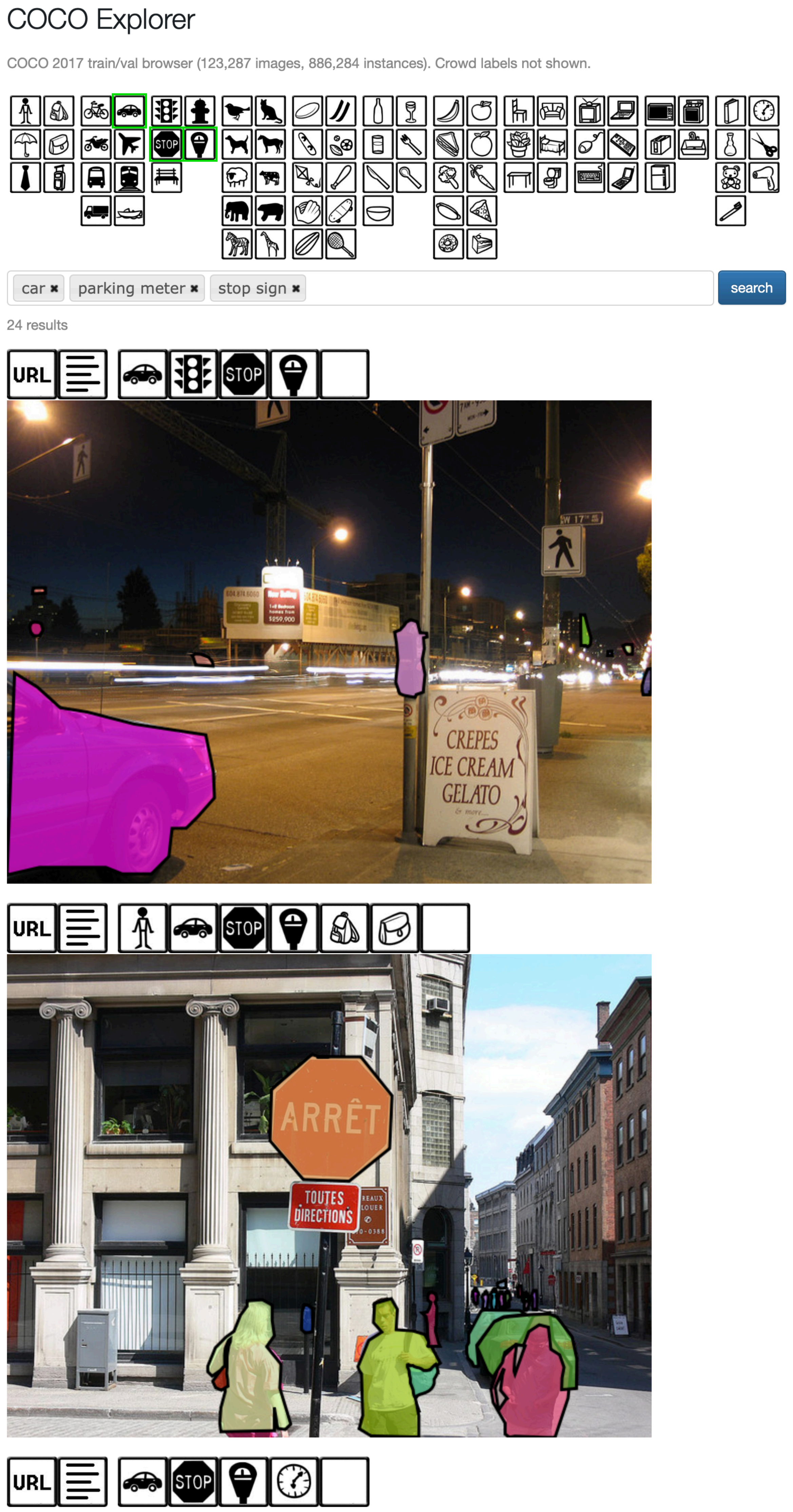
https://keras.io/applications/

https://www.tensorflow.org/api_docs/python/tf/keras/applications

Common Datasets in Computer Vision

http://cocodataset.org/#home

- **Microsoft Coco (common objects in context):**
  - 330,000 images, 80 categories
  - Labeling for segmentation and object detection



Common Datasets in Computer Vision

Youtube-8M/#home

YouTube-8M Segments Dataset

237K Human-verified Segment Labels	1000 Classes	5.0 Avg. Segments / Video

In addition to annotating the topical entity of the full-video, we want to understand when the entity occurs in videos. Given a 5-second segment and a query class, our human raters are asked to verify whether the entity is identified within the segment. To speed up the annotation process, our human raters do not report presence or absence of non-query classes.

Datasets for Speech/Audio

Librivox

1000 hours of speech and transcripts taken from free on-line audio book website

TIMIT

speech + transcript

Linguistic Data Consortium (LDC)

some free, some $$$

Datasets for Speech/Audio

Libravox

1000 hours of speech and transcripts taken from free on-line audio book website

Linguistic Data Consortium (LDC)

speech + transcript

(some free, some $$$)

TIMIT

Google's audioset

audio events (tagged sounds)

Other Sources of Data

Kaggle datasets for on-line ML competitions

UCI ML Archive

Google Dataset Search

Amazon Web Services Open Data