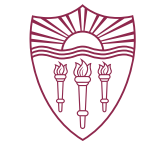
USC Viterbi
School of Engineering



# WORKING WITH DATA

## EE 541 – UNIT 9

DR. BRANDON FRANZKE

Fall 2026

# OUTLINE

- 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



# DESIGNING DATASETS

# 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")

# 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

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

where $W$ is the total number of free parameters (i.e., synaptic weights and biases) in the network, $\varepsilon$ 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 $\varepsilon$ 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 ~ 6,000 trainable parameters!

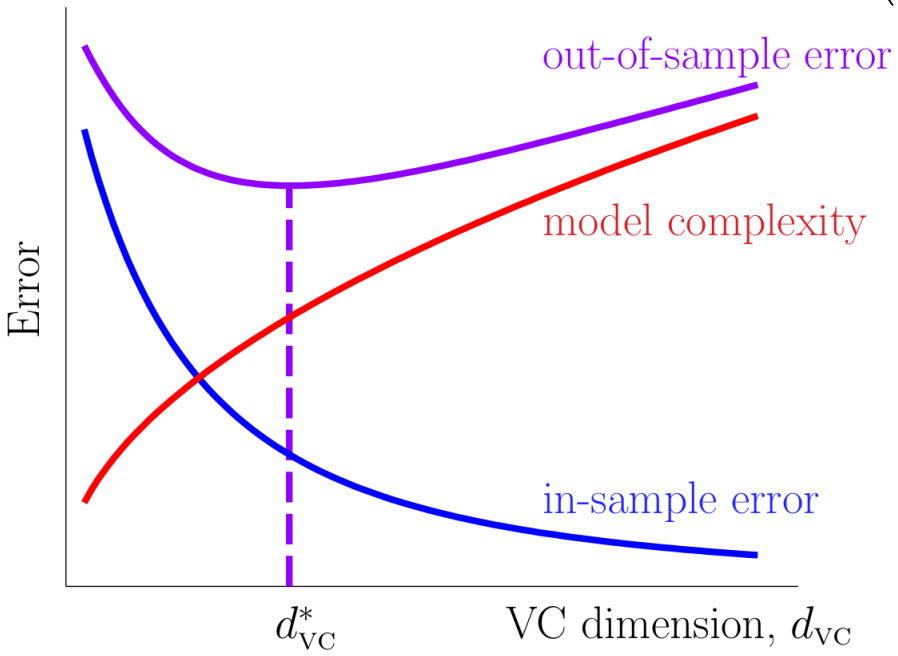
Obviously, this is a big-O rule of thumb!

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

# HOW MUCH DATA IS NEEDED (GENERAL)?

## Vapnik-Chervonenkis (VC) Dimension

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

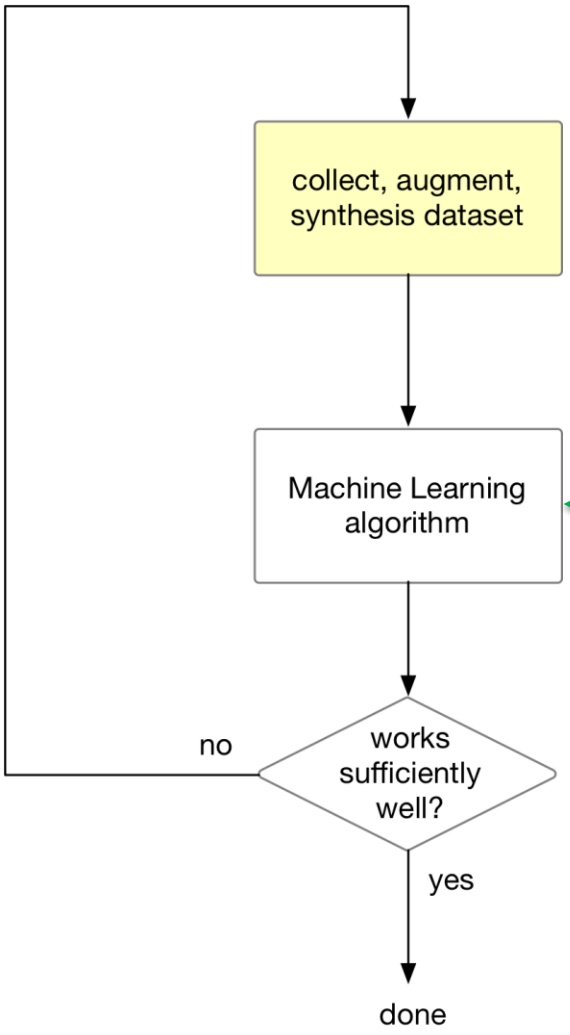



VC Dimension difficult to compute for complex (deep learning models)

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

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

# 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** | find data online, crowdsource, scrape online resources |
| **Labeling** | labor intensive task<br>(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 benchmarks

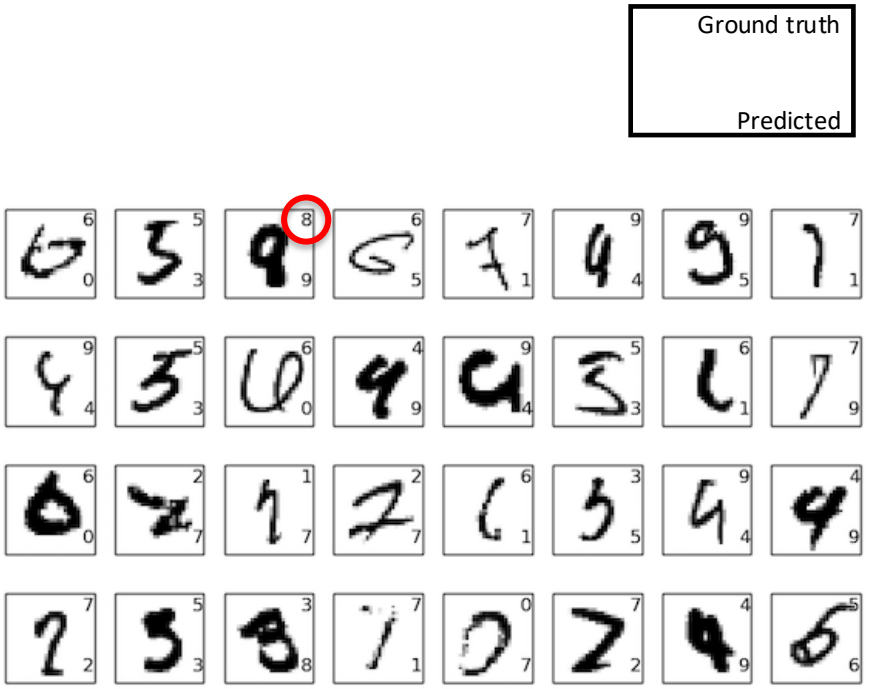
in our class, we are crowd-sourcing project datasets to try to illustrate the practical issues, but still not at a practical scale

# DATASET CONTAMINATION / CLEANING

many datasets contain mislabeled or ambiguous data



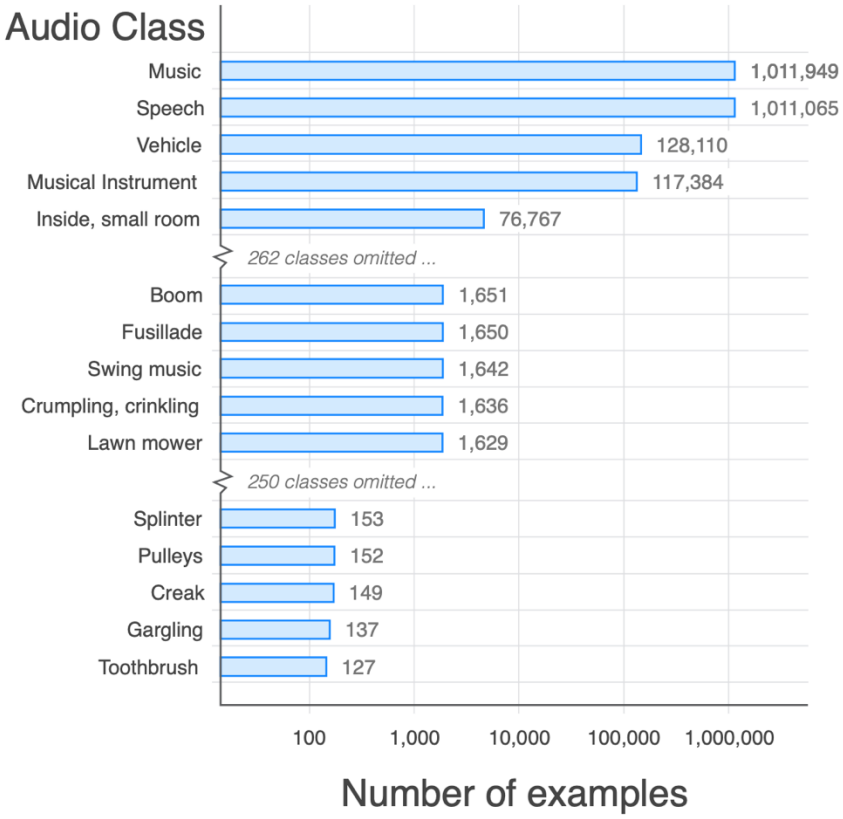
"cat?" "dog?"



# 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**

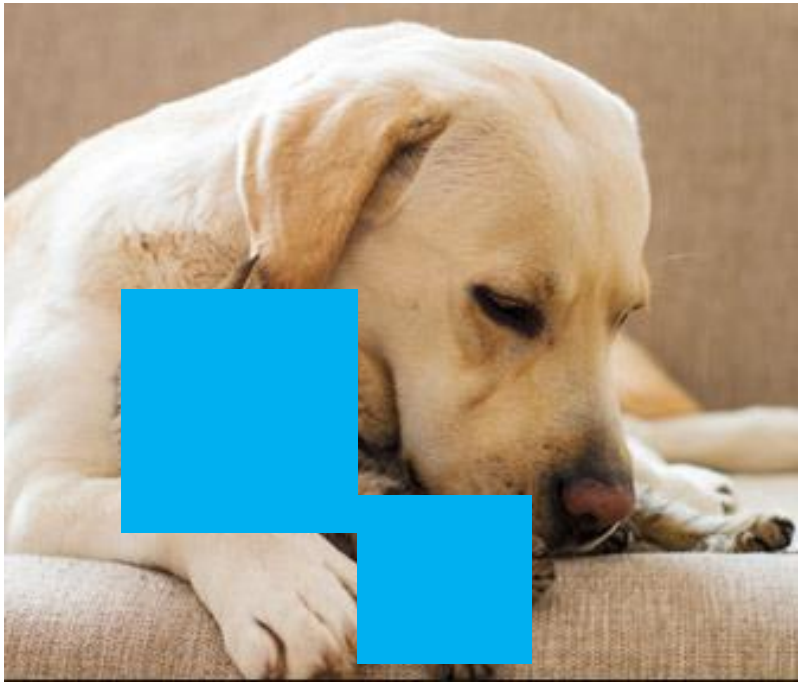
# DATASET CONTAMINATION / CLEANING

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?

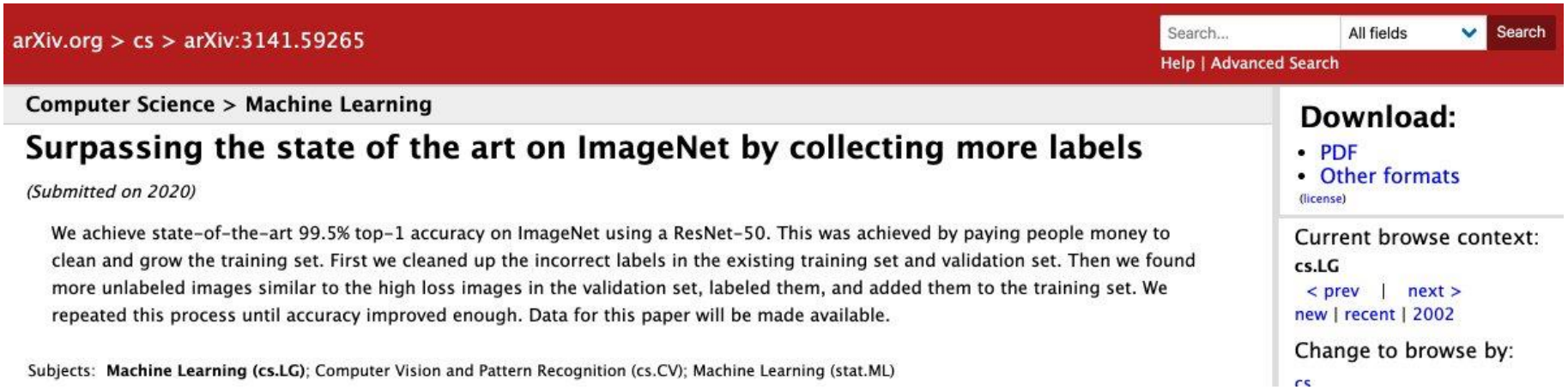
Fill-in for missing data

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




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

"cut-out"
(remove patches)

translate

add noise

camera
modeling

blur (change resolution)

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

# 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



# DEEP LEARNING WORKFLOW

# 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

keep your training simple to start

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

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

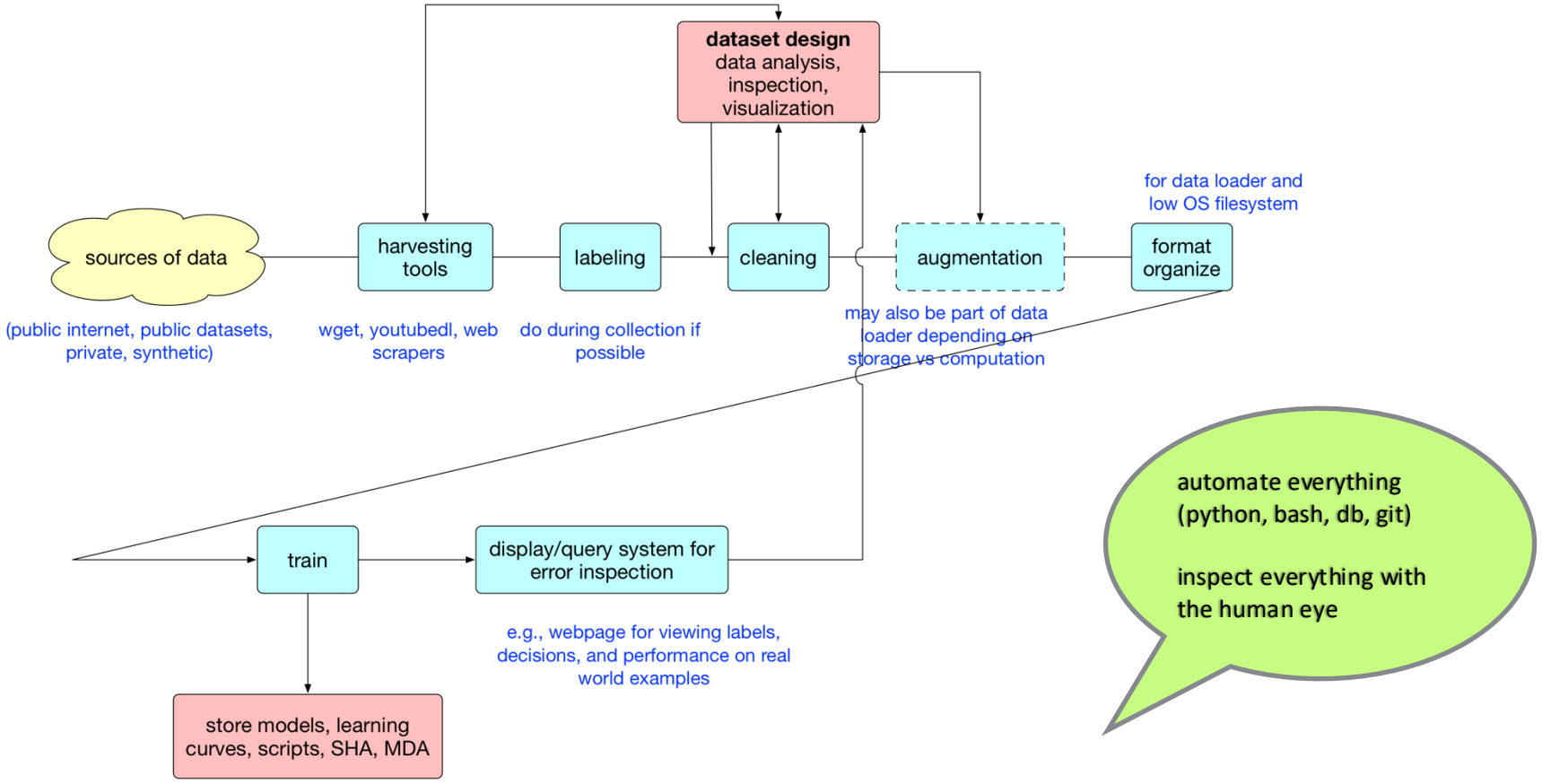
viewing/interpreting datasets and examples

cultivating/updating your dataset

training with version control and auto-documentation

testing and visualization of the result on real-world data

# TYPICAL FLOW FOR DEEP LEARNING DEVELOPMENT




popular blog by Tesla 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



# DATA NORMALIZATION

# 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



# COMMON DATA NORMALIZATION METHODS

recall: data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \qquad \mathbf{X}^T = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_N]$$

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

$$\hat{\mathbf{R}}_\mathbf{x} = \langle \mathbf{x}\mathbf{x}^T \rangle_\mathcal{D} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$$

$$= \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

$$\hat{\mathbf{K}}_\mathbf{x} = \hat{\mathbf{R}}_\mathbf{x} - \hat{\mathbf{m}}_\mathbf{x} \hat{\mathbf{m}}_\mathbf{x}^T$$

$$= \langle [\mathbf{x} - \hat{\mathbf{m}}_\mathbf{x}][\mathbf{x} - \hat{\mathbf{m}}_\mathbf{x}]^T \rangle_\mathcal{D}$$

$$= \frac{1}{N} \sum_{n=1}^{N} [\mathbf{x}_n - \hat{\mathbf{m}}_\mathbf{x}][\mathbf{x}_n - \hat{\mathbf{m}}_\mathbf{x}]^T$$

# COMMON DATA NORMALIZATION METHODS

## feature-wise standardization

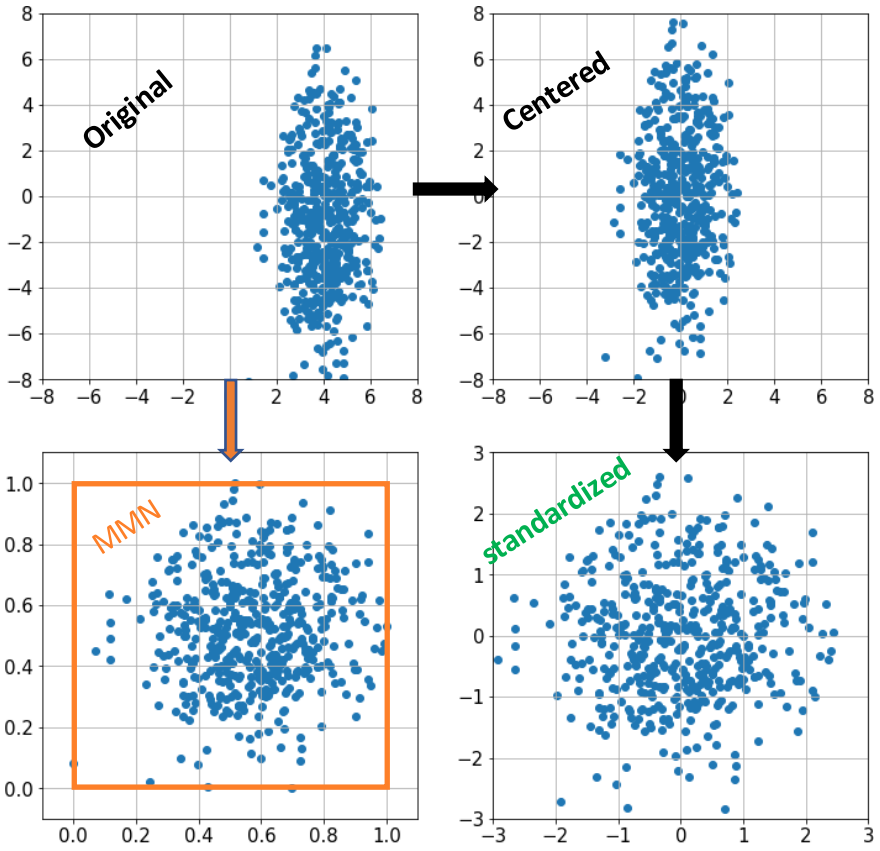
$$v_n[i] = \frac{x_n[i] - \hat{m}_x[i]}{\hat{\sigma}_x[i]}$$

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]



# 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

# KL-EXPANSION

$$\boldsymbol{K}_X \boldsymbol{e}_k = \lambda_k \boldsymbol{e}_k \quad k = 1, 2, \ldots, D$$ *Eigen equation*

$$\boldsymbol{e}_k^T \boldsymbol{e}_l = \delta[k-l], \qquad \lambda_k \geq 0$$ *orthonormal eigen vectors*

$$\boldsymbol{X}(t) = \sum_{k=1}^{D} X_k(t) \boldsymbol{e}_k$$ *change of coordinates*

$$X_k(t) = \boldsymbol{e}_k^T \boldsymbol{X}(t)$$

$$\mathbb{E}\{X_k(t) X_l(t)\} = \boldsymbol{e}_k^t \boldsymbol{K}_X \boldsymbol{e}_l = \lambda_k \delta[k-l]$$ *uncorrelated components*

$$\boldsymbol{K}_X = \sum_{k=1}^{D} \lambda_k \boldsymbol{e}_k \boldsymbol{e}_k^t = \boldsymbol{E} \boldsymbol{\Lambda} \boldsymbol{E}^T$$ *Spectral Theorem*

$$\mathbb{E}\{\|\boldsymbol{X}(t)\|^2\} = \mathrm{tr}(\boldsymbol{K}_x) = \sum_{k=1}^{D} \lambda_k$$ *Total Energy*

Always exists because $\boldsymbol{K}_X$ is symmetric and positive semi- definite (PSD)

# KARHUNEN–LOÈVE (KL) EXPANSION

**Can always find orthonormal set of e-vectors of $\mathbf{K}$**

These are alternate coordinate systems (rotations, reflections)

in this eigen-coordinate system, the components are uncorrelated

*"principal components"*

The eigen-values are the variance (energy) in each principal direction

*(reduce dimensions by "throwing out" low-energy components)*

# KL-EXPANSION

$$d_k(t) = \mathbf{e}_k^T \mathbf{x}(t) \qquad k = 1, 2, \ldots, D$$

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

$$\mathbf{K_d} = \mathbf{E}^T \mathbf{K_x} \mathbf{E}$$

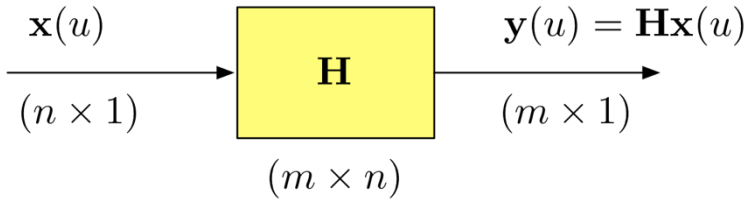
$$= \mathbf{E}^T (\mathbf{E} \boldsymbol{\Lambda} \mathbf{E}^T) \mathbf{E}$$

$$= \boldsymbol{\Lambda} = \mathbf{diag}(\lambda_k)$$

**Multiplying data by $E^T$ makes the components uncorrelated**

$$\mathbf{E} = [\mathbf{e}_1 \quad | \quad \mathbf{e}_2 \quad | \quad \mathbf{e}_3 \quad | \quad \cdots \quad | \quad \mathbf{e}_D]$$

# RANDOM VECTORS




$$\mathbf{m_y} = \mathbf{H}\mathbf{m_x}$$

$$\mathbf{R_y} = \mathbf{H}\mathbf{R_X}\mathbf{H}^T$$

$$\mathbf{K_y} = \mathbf{H}\mathbf{K_X}\mathbf{H}^T$$

## Special case

$$y(t) = \mathbf{b}^T\mathbf{x}(t) \qquad (1\times1)$$

$$m_y = \mathbf{b}^T\mathbf{m_x}$$

$$\mathbb{E}\{y^2(t)\} = \mathbf{b}^T\mathbf{R_x}\mathbf{b}$$

$$\sigma_y^2 = \mathbf{b}^T\mathbf{K_x}\mathbf{b}$$

## example math

$$\begin{aligned} R_y &= \mathbb{E}\{y(t)y^T(t)\} \\ &= \mathbb{E}\{(\mathbf{H}\mathbf{x}(t))(\mathbf{H}\mathbf{x}(t))^T\} \\ &= \mathbb{E}\{\mathbf{H}\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{H}^T\} \\ &= \mathbf{H}\mathbb{E}\{\mathbf{x}(t)\mathbf{x}^T(t)\}\mathbf{H}^T \\ &= \mathbf{H}\mathbf{R_X}\mathbf{H}^T \end{aligned}$$

Note that covariance/correlation matrices are symmetric, positive semi-definite

# KL-EXPANSION - RELATION TO WHITENING

$$w_k(t) = \frac{X_k(t)}{\sqrt{\lambda_k}} = \frac{\mathbf{e}_k^T \mathbf{x}(t)}{\sqrt{\lambda_k}} \qquad k = 1, 2, \ldots, D$$

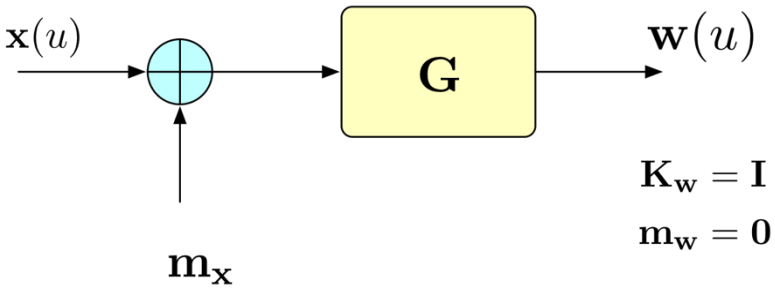
$$\mathbf{w}(t) = \boldsymbol{\Lambda}^{-1/2} \mathbf{E}^T \mathbf{x}(\mathrm{t})$$

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

requires every $\lambda_k > 0$

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

$$\mathbf{G} = \mathbf{U} \boldsymbol{\Lambda}^{-1/2} \mathbf{E}^T$$




$$\mathbf{K_x} = \mathbf{H}\mathbf{H}^T \quad \Rightarrow \quad \mathbf{G} = \mathbf{H}^{-1}$$



# 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

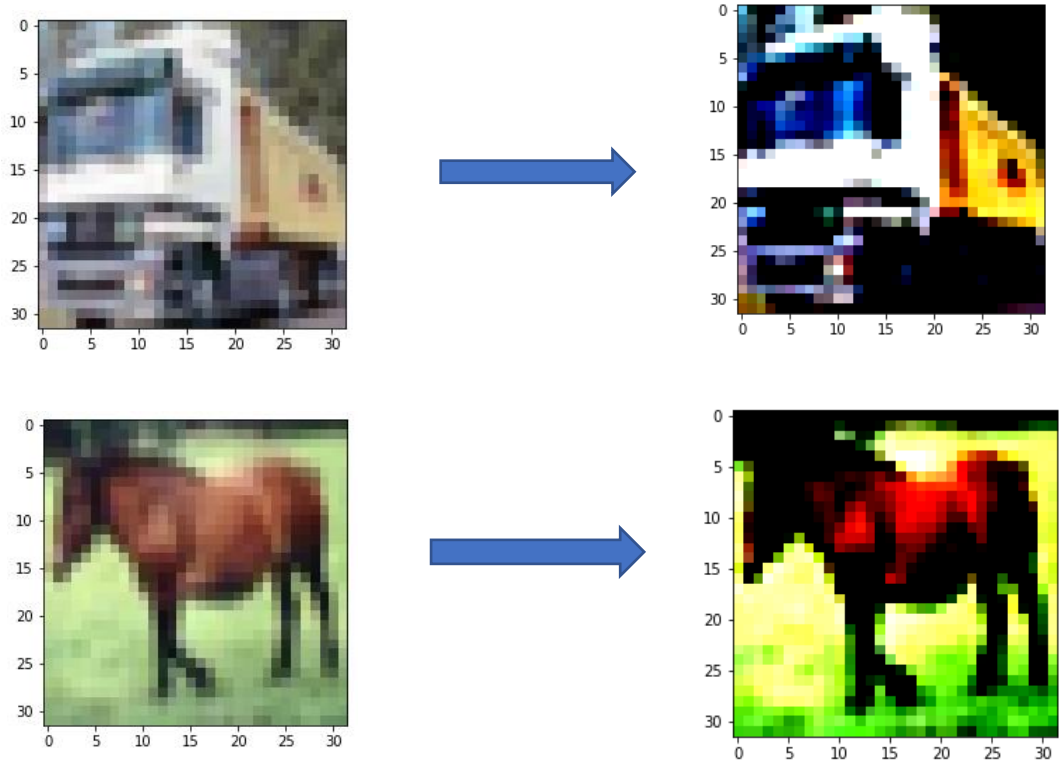
all feature vector components are equally important

aka: ZCA ("zero-phase component analysis") when $G = E\Lambda^{(-1/2)}E^T$

# DATA NORMALIZATION - GLOBAL CONTRAST NORMALIZATION

$x$: Single Image

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



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

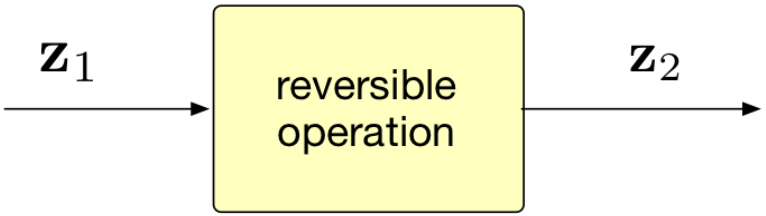
# OUTLINE FOR SLIDES

- 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



# DIMENSIONALITY REDUCTION

# THEOREM OF REVERSIBILITY




$z_2$ can be used to perform inference on $d$ without a loss of information

This may complicate or simplify, but for example, the Max-Likelihood processing will yield same results

# THEOREM OF IRRELEVANCE

Suppose we have two observations

$$p(\mathbf{z}_1, \mathbf{z}_2|d) = p(\mathbf{z}_1|\mathbf{z}_2, d)p(\mathbf{z}_2|d)$$

If the following holds

$$p(\mathbf{z}_1|\mathbf{z}_2, d) = p(\mathbf{z}_1|\mathbf{z}_2)$$

Then we say that $z_1$ is irrelevant given $z_2$ for the purposes of inferring $d$ (i.e., detection/estimation)

This means that $z_1$ can be thrown away if we have $z_2$ without loss of information for the purposes of inferring $d$

**some irreversible operations are ok!**

# SET OF SUFFICIENT STATISTICS

A set of sufficient statistics for inferring $d$ is a function of the observations that makes the observations irrelevant (redundant)

$$p(\mathbf{z}, \mathbf{g}(\mathbf{z})|d) = p(\mathbf{z}|\mathbf{g}(\mathbf{z}), d)p(\mathbf{g}(\mathbf{z})|d)$$

$$= p(\mathbf{z}|\mathbf{g}(\mathbf{z}))p(\mathbf{g}(\mathbf{z})|d)$$

$$\equiv p(\mathbf{g}(\mathbf{z})|d)$$

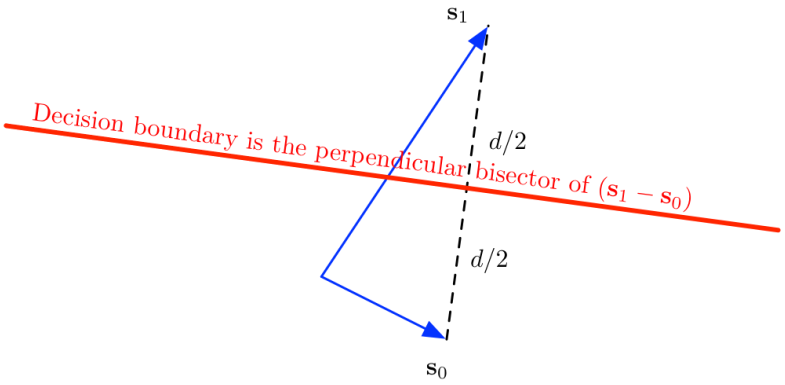
Example:

$\{\mathbf{z}^{\mathrm{t}}\mathbf{s}_m\}_{m=0}^{M-1}$ is a set of sufficient stats for the vector AWGN channel

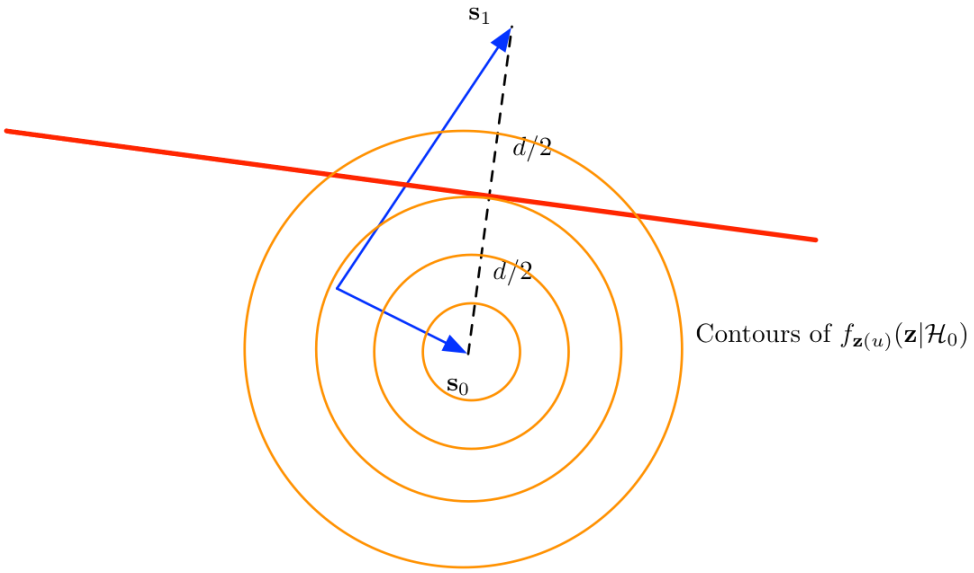
**Note:** this can be a large reduction in the number of dimensions without any loss of optimality

sufficient statistics are ideal features

# EXAMPLE: BINARY MAP DECISIONS (EQUAL PRIORS)




$$(\mathbf{s}_1 - \mathbf{s}_0)^t \mathbf{z} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2}$$




Error probability given hypothesis 0 is the probability that noise throws observation over decision boundary

# SUFFICIENT STAT FOR THIS PROBLEM…

$$g(\mathbf{z}) = \mathbf{z}^{\mathrm{t}}(\mathbf{s}_1 - \mathbf{s}_0)$$

definitely not reversible!

**Note:** the original dimension may be 2 or 1,000,000 and the sufficient stat still has only dimension 1.

this reduces the observation to the essential information relevant to inferring between the two possibilities from the observation

(aside: the performance is not a function of dimension either in this case)

sufficient statistics are **ideal** features

# KL-EXPANSION - RELATION TO PCA

$$\tilde{x}_k(t) = \mathbf{e}_k^T \mathbf{x}(t) \qquad k = 1, 2, \ldots, T$$

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

$$\mathbf{K}_{\tilde{\mathbf{x}}} = \mathbf{\Lambda}_{[:T]} \qquad \text{assumes ordered e-values } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

$$\mathbb{E}\{\|\tilde{\mathbf{x}}(t)\|^2\} = \sum_{k=1}^{T} \lambda_k$$

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

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

$$\mathbf{E}_{[:T]} = [\mathbf{e}_1 \quad | \quad \mathbf{e}_2 \quad | \quad \mathbf{e}_3 \quad | \quad \cdots \quad | \quad \mathbf{e}_T]$$

# KL/PCA FOR DATA

Everything is the same, except we use data-averaging instead of $\mathbb{E}[\cdot]$

$$\widehat{\mathbf{R}}_{\mathbf{x}} = \langle \mathbf{x}\mathbf{x}^T \rangle_{\mathcal{D}}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T$$

$$= \frac{1}{N} \mathbf{X}^T \mathbf{X}$$

**Both KL/PCA can be applied to $R$ or $K$.**
**Center $x$ if you want to use $K$**
**$x \leftarrow x - m$**
**(same if mean is zero)**

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \qquad \mathbf{X}^T = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_N]$$

"stacked" data matrix

# KL/PCA FOR DATA

## PCA for data

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

$\mathbf{E}_{[:T]} = [\mathbf{e}_1 \mid \mathbf{e}_2 \mid \mathbf{e}_3 \mid \cdots \mid \mathbf{e}_T]$

apply to the "stacked" data matrix

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

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

$$\underset{T\times T}{\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}}}$$

dimension reduced from $D$ to $T$

$$\widetilde{\mathbf{X}}^T = \left[\mathbf{E}_{[:T]}^{\mathrm{T}} \mathbf{x}_0 \quad \mathbf{E}_{[:T]}^{\mathrm{T}} \mathbf{x}_1 \quad \cdots \quad \mathbf{E}_{[:T]}^{\mathrm{T}} \mathbf{x}_{\mathrm{N}}\right] = \mathbf{E}_{[:T]} \mathbf{X}^T$$

# KL/PCA FOR DATA — RELATION TO SVD

## SVD for an arbitrary matrix $\mathbf{A}$

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times m}{\mathbf{U}}\ \underset{m\times n}{\mathbf{\Sigma}}\ \underset{n\times n}{\mathbf{V}^T}$$

$\mathbf{U}$, $\mathbf{V}$ are orthogonal matrices, $\mathbf{\Sigma}$ is "diagonal" with singular values on diagonal

Use SVD to expand matrix $\mathbf{A}^T\mathbf{A}$

$$\begin{aligned}\mathbf{A}^T\mathbf{A} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \underset{n\times n}{\mathbf{V}}\ \underset{n\times n}{\mathbf{\Sigma}^T\mathbf{\Sigma}}\ \underset{n\times n}{\mathbf{V}^T} \\ &= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T\end{aligned}$$

The SVD for $A$ provides the KL factorization for the non-negative definite symmetric matrix $\mathbf{A}^T\mathbf{A}$

# KL/PCA FOR DATA — RELATION TO SVD

## SVD for stacked data matrix $\mathbf{X}$

$$\underset{N\times D}{\mathbf{X}} = \underset{N\times N}{\mathbf{U}}\ \underset{N\times D}{\mathbf{\Sigma}}\ \underset{D\times D}{\mathbf{V}^T}$$

$$\underset{D\times D}{\mathbf{X}^T\mathbf{X}} = \underset{D\times D}{\mathbf{V}}\ \underset{D\times D}{\mathbf{\Sigma}^T\mathbf{\Sigma}}\ \underset{D\times D}{\mathbf{V}^T}$$

$$= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$$

Equivalent approaches:

1) Find SVD of $\mathbf{X}$, take $\mathbf{V}$

2) Find Eigen decomposition of $\mathbf{X}^T\mathbf{X}$, take $\mathbf{V} = \mathbf{E}$

3) Find SVD of $\mathbf{X}^T\mathbf{X}$, take $\mathbf{V} = \mathbf{U} = \mathbf{E}$

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

# KL/PCA FOR DATA — RELATION TO SVD

**Equivalent approaches:**

1) Find SVD of $\mathbf{X}$, take $\mathbf{V}$

2) Find Eigen decomposition of $\mathbf{X}^T\mathbf{X}$, take $\mathbf{V} = \mathbf{E}$

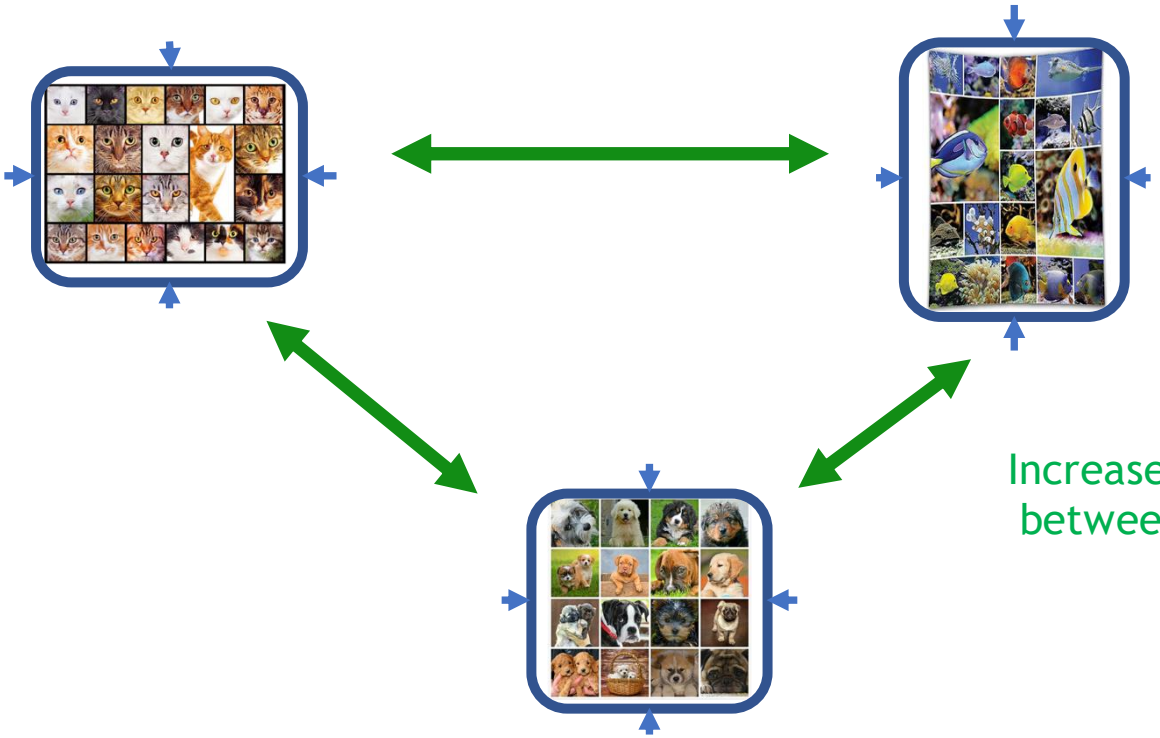
3) Find SVD of $\mathbf{X}^T\mathbf{X}$, take $\mathbf{V} = \mathbf{U} = \mathbf{E}$

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

Note: using method 3 (with `numpy.linalg.svd`) instead of method 2 (with `numpy.linalg.eig`) returns the e-vectors in sorted order and `eig` does not

# LINEAR DISCRIMINANT ANALYSIS (LDA)

Like PCA but is supervised and for classification problem



Decrease variance within classes

Increase variance between classes

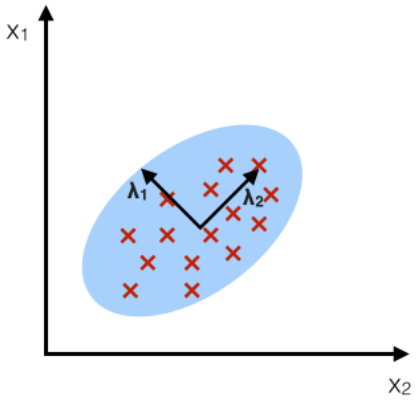
LDA keeps features which can discriminate well between classes

good reference is this blog

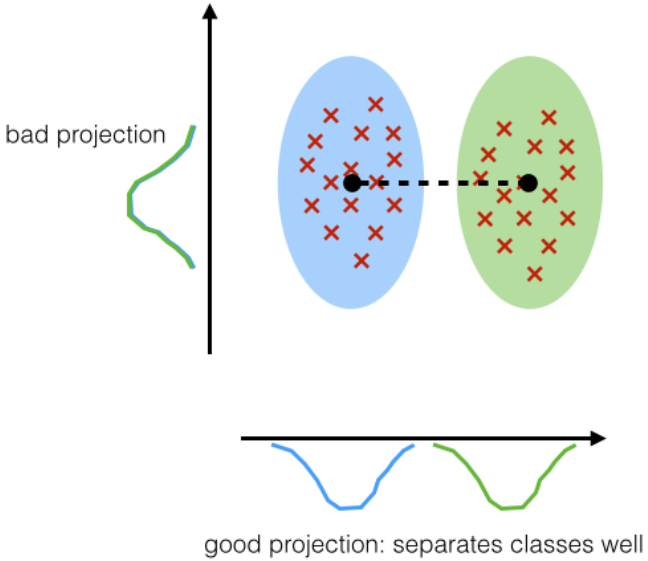
# LINEAR DISCRIMINANT ANALYSIS (LDA)

Like PCA but is supervised and for classification problem

**PCA:**
component axes that maximize the variance




**LDA:**
maximizing the component axes for class-separation




good reference is this blog



# PRE-PROCESSING (PCA, LDA, NORMALIZATION)

Use statistics collected from **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



# FINDING DATA

# 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" means there is a dataset/dataloader and the framework automates downloading and loading

# DATASETS AVAILABLE IN PYTORCH

## MNIST (MLP / CNN):

- 28x28 images, 10 classes
- Initial benchmark
- *SOTA testacc: >99%*

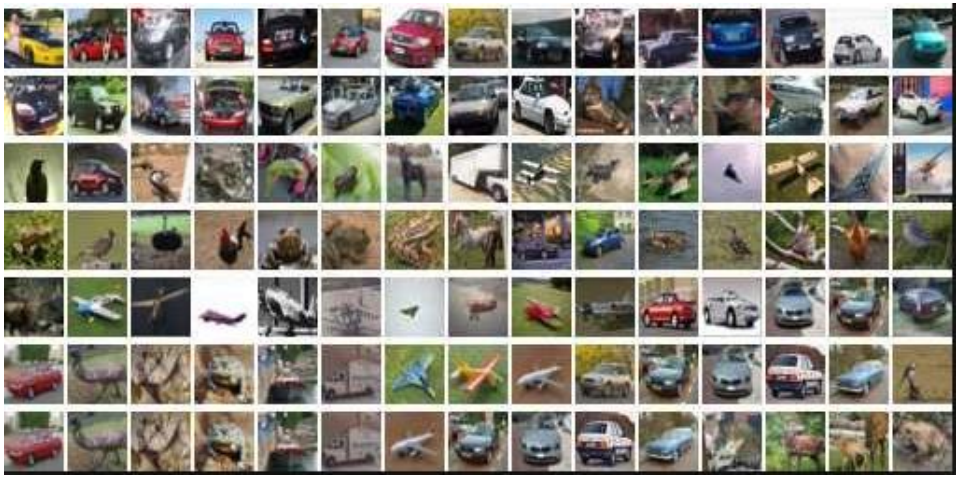


## Fashion MNIST (MLP / CNN):

- 28x28 images, 10 classes
- More challenging than MNIST, but same parameters
- *SOTA testacc: 95%*

## CIFAR-10, -100 (CNN):

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



# EXTERNAL DATASETS IN PYTORCH

## 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

## Boston housing price

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

```
import torch
from torchtext import data
from torchtext import datasets

TEXT = data.Field()
LABEL = data.LabelField()

train_data, test_data = datasets.IMDB.splits(TEXT, LABEL)

train_data, valid_data = train_data.split()
```

## AG News topics classification

- 127,600 articles from AG News (120,000 train, 7,600 test)
- labeled with 4 topics.
- each article: a title and a short description

# COMMON DATASETS IN COMPUTER VISION

## ImageNet:

- 1.28M training images, 1000 classes (the ILSVRC subset of ImageNet's 14M)
- Common benchmark for image classification
- *SOTA testacc: >90%*

PyTorch has many of the popular image classification networks already available and pre-trained on ImageNet

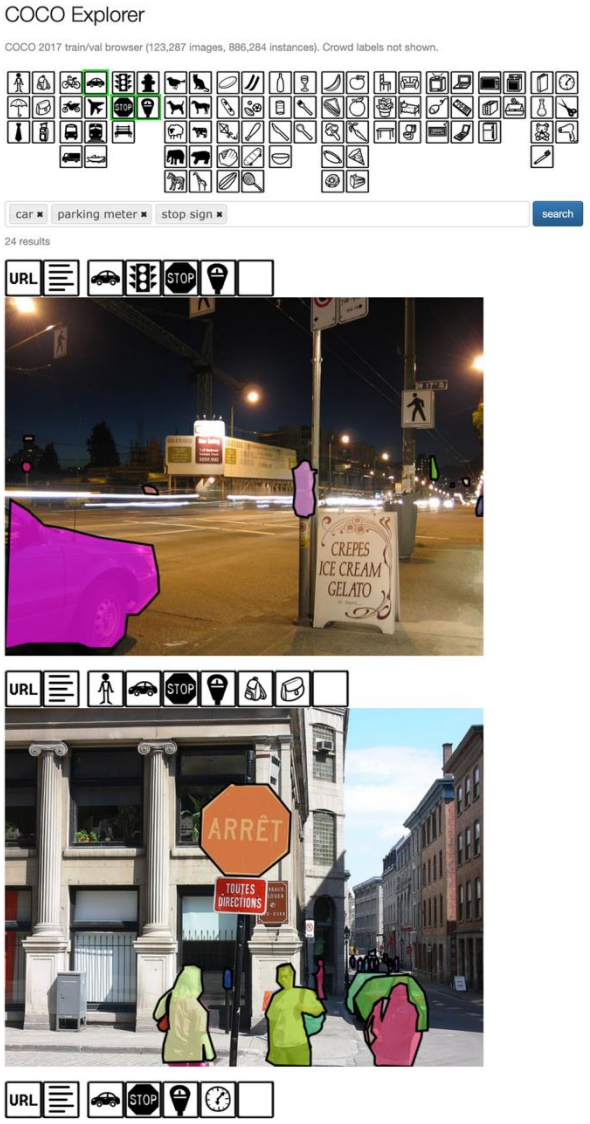
| Model | Size | Top-1 Accuracy | Top-5 Accuracy | Parameters | Depth |
|---|---|---|---|---|---|
| Xception | 88 MB | 0.790 | 0.945 | 22,910,480 | 126 |
| VGG16 | 528 MB | 0.713 | 0.901 | 138,357,544 | 23 |
| VGG19 | 549 MB | 0.713 | 0.900 | 143,667,240 | 26 |
| ResNet50 | 98 MB | 0.749 | 0.921 | 25,636,712 | - |
| 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://pytorch.org/docs/stable/torchvision/models.html

# COMMON DATASETS IN COMPUTER VISION

http://cocodataset.org




## Microsoft Coco

## (common objects in context):

- 330,000 images
- 80 categories
- Labeling for segmentation and object detection

# COMMON DATASETS IN COMPUTER VISION

Youtube-8M

## YouTube-8M Segments Dataset

| 237K | 1000 | 5.0 |
| --- | --- | --- |
| Human-verified Segment Labels | Classes | 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

LibriSpeech: 1000 hours of read speech and transcripts from LibriVox audiobooks

## 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