# Copyright Notice

DeepLearning.AI

# Advanced Labeling, Augmentation and Data Preprocessing

## Welcome

DeepLearning.AI

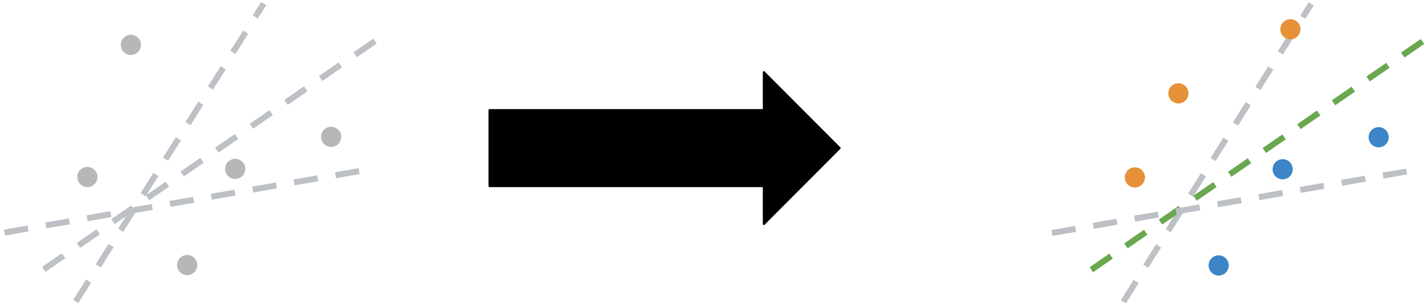
Advanced Labeling

# Semi-supervised Labeling

# Outline

- Overview of advanced labeling techniques:
  - Semi-supervised learning
  - Active learning
  - Weak supervision with Snorkel

# Why is Advanced Labeling Important?



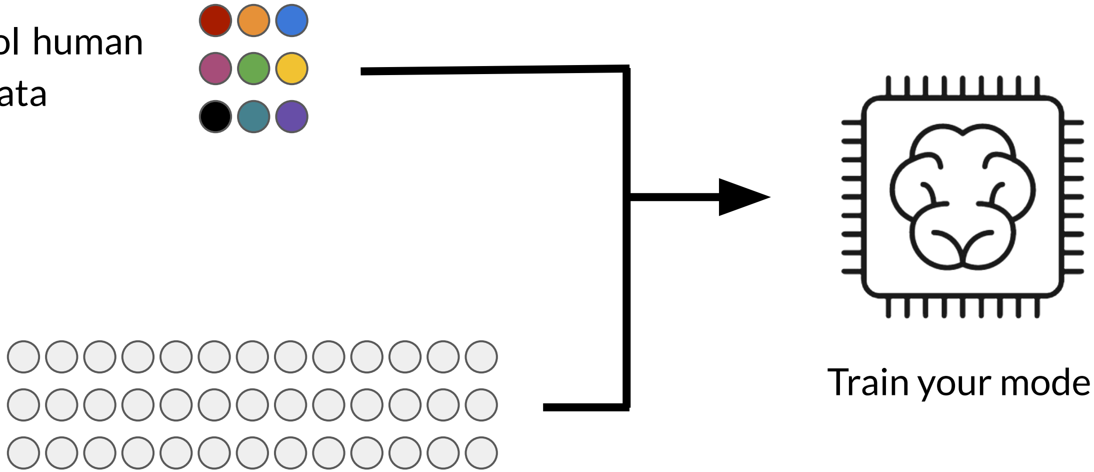
ML use is growing everywhere

...as is the need for labeled training sets

- Manually labeling of data is expensive
- Unlabeled data is usually cheap and easy to get
- Unlabeled data contains a lot of information that can improve our model

# Human labeling, Semi-supervised

Small pool human labeled data

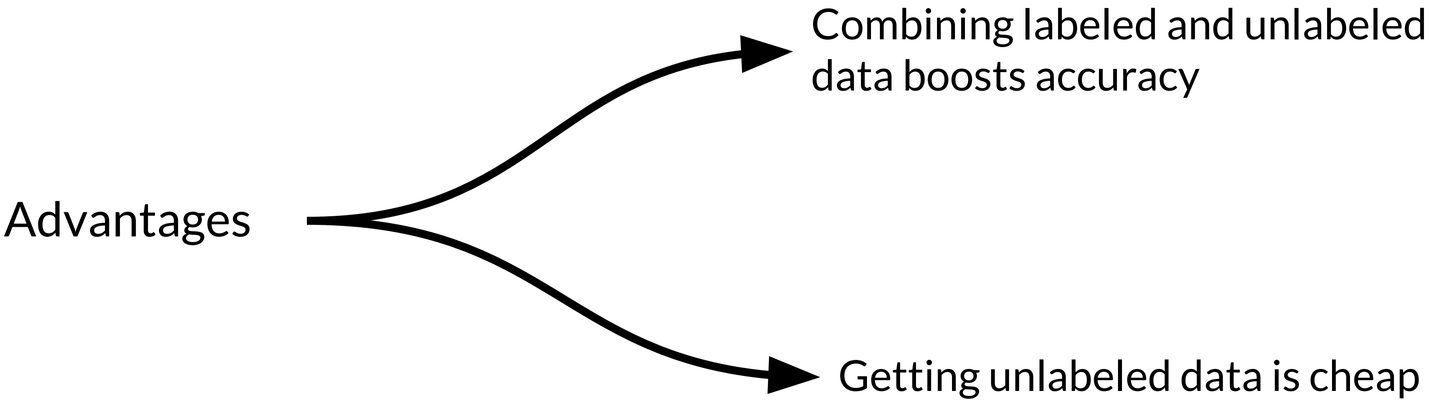


Large pool unlabeled data

Train your model

Relies on some degree of uniformity or clustering within feature space

# Human labeling, Semi-supervised



Advantages

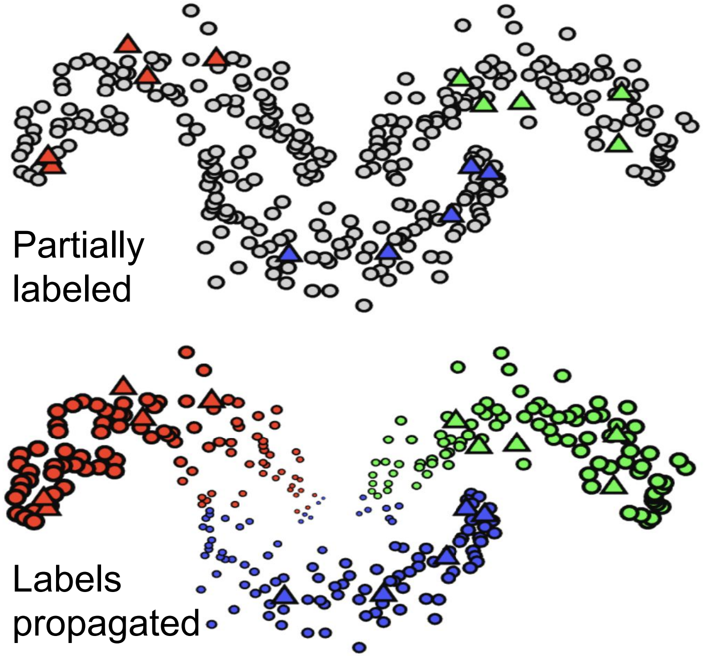
Combining labeled and unlabeled data boosts accuracy

Getting unlabeled data is cheap

# Label propagation

- Semi-supervised ML algorithm
- A subset of the examples have labels
- Labels are propagated to the unlabeled points:
    - Based on similarity or “community structure”

# Label propagation - Graph based



Unlabeled examples can be assigned labels based on their neighbors

DeepLearning.AI

## Advanced Labeling

# Active Learning

# Active learning

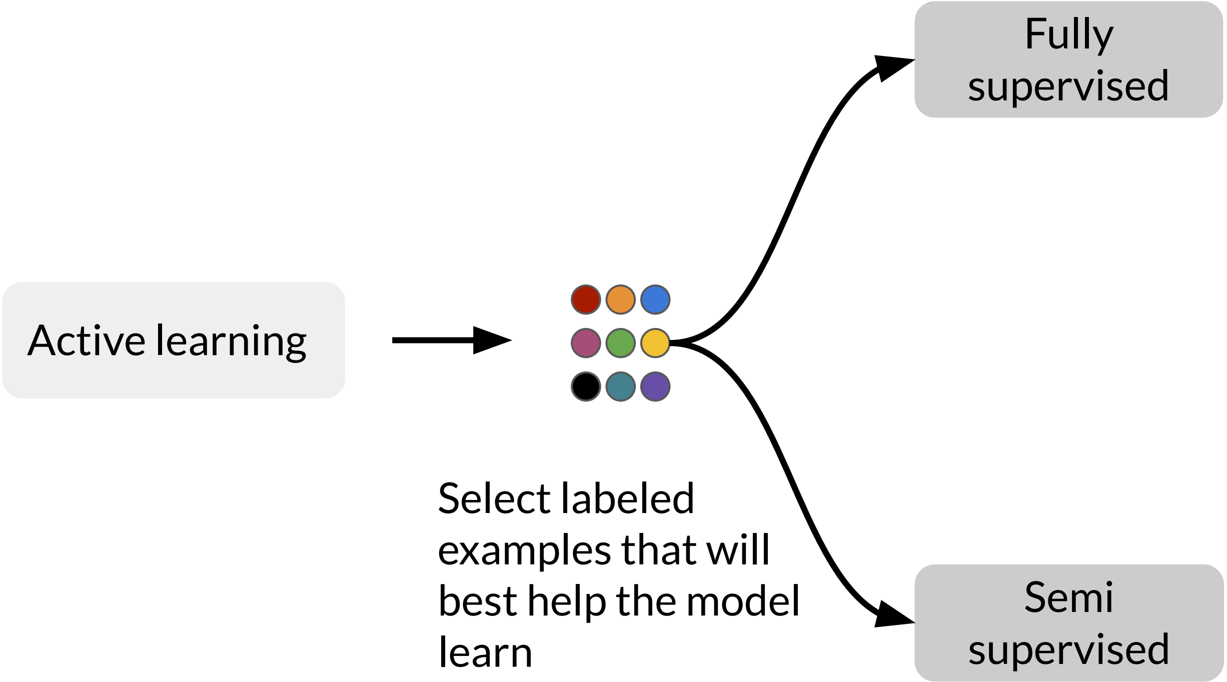
- A family of algorithms for intelligently sampling data
- Select the points to be labeled that would be most informative for model training
- Very helpful in the following situations:

Constrained data budgets: you can only afford labeling a few points

Imbalanced dataset:  helps selecting rare classes for training

Target metrics:  when baseline sampling strategy does not improve selected metrics

# Active learning strategies



Active learning

Select labeled examples that will best help the model learn

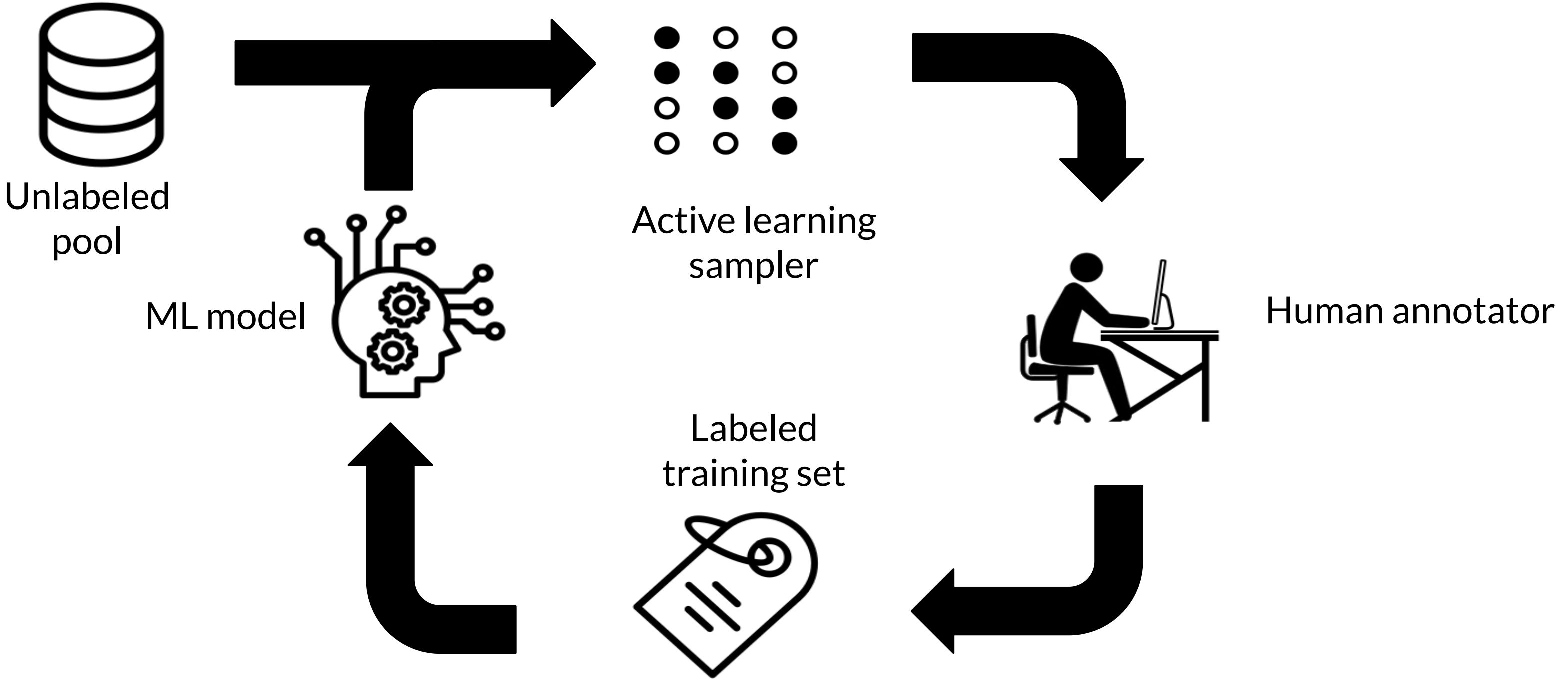
Fully supervised

Form a training dataset with only those examples

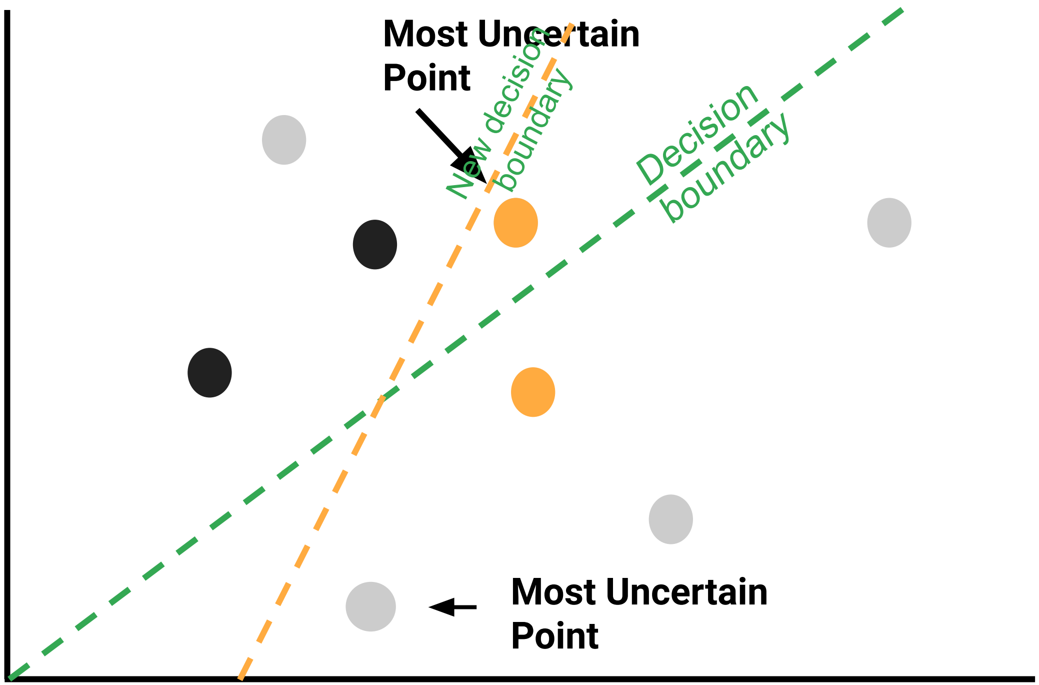
Semi supervised

Label propagation for unlabeled examples

# Active learning cycle

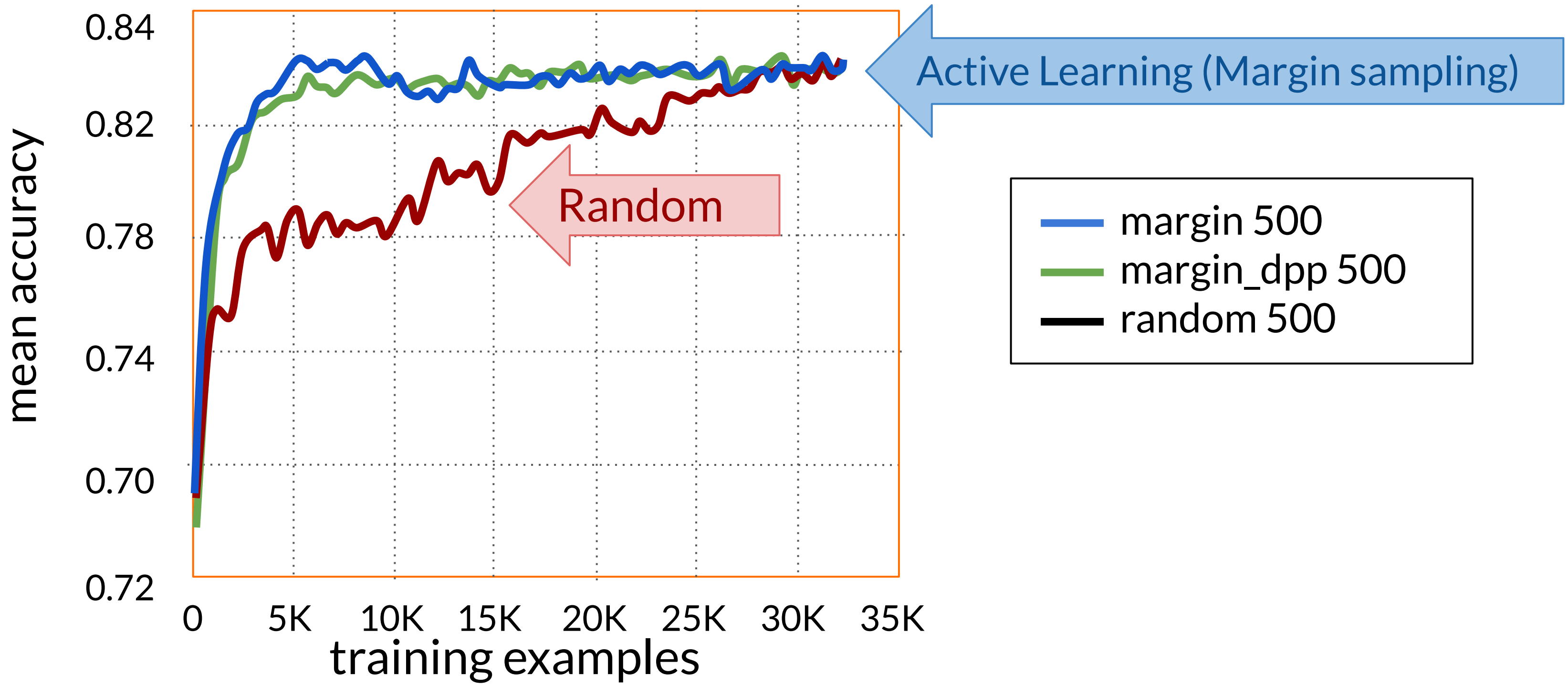


# Margin sampling



| | |
|---|---|
| ● | Class 1 |
| ● | Class 2 |
| ● | Unlabeled Point |

# Example results - Different Sampling Techniques



# Active learning sampling techniques

**Margin sampling**: Label points the current model is least confident in.

**Cluster-based sampling**: sample from well-formed clusters to "cover" the entire space.

**Query-by-committee:** train an ensemble of models and sample points that generate disagreement.

**Region-based sampling:** Runs several active learning algorithms in different partitions of the space.

DeepLearning.AI

# Advanced Labeling

## Weak Supervision

# Hand labeling: intensive labor

"Hand-labeling training data for machine learning problems is effective, but very labor and time intensive.  This work explores how to use algorithmic labeling systems relying on other sources of knowledge that can provide many more labels but which are noisy."

Jeff Dean, March 14, 2019

# Weak supervision

“Weak supervision is about leveraging higher-level and/or noisier input from subject matter experts (SMEs).”

-Weak Supervision: The New Programming Paradigm for Machine Learning
Blog post by Ratner, Varma, Hancock, Re, and Hazy Lab

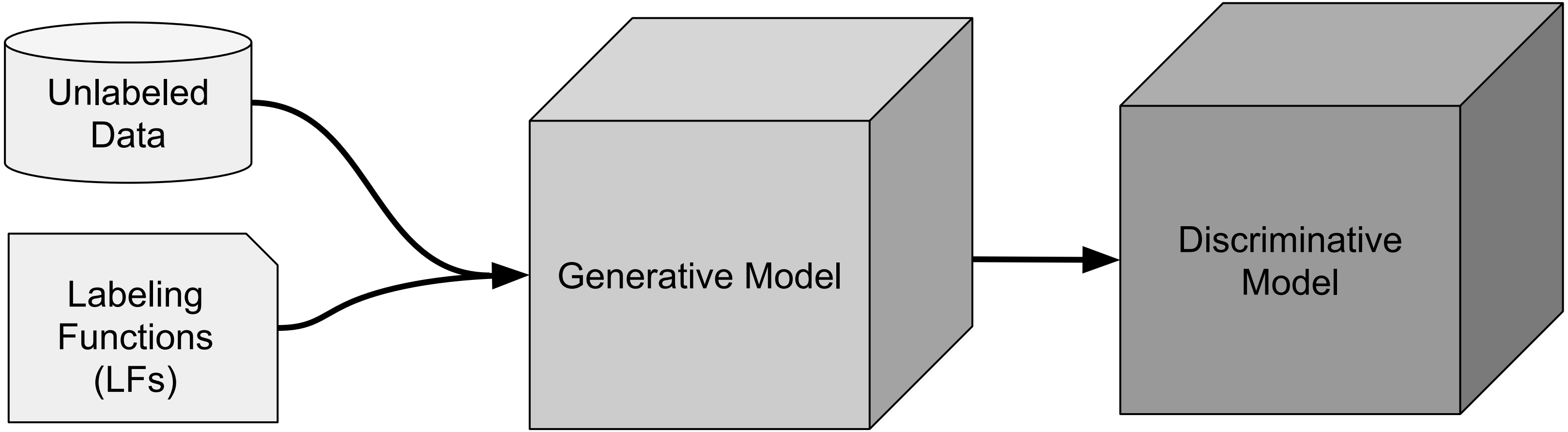
# Weak supervision

- Unlabeled data, without ground-truth labels
- One or more weak supervision sources
  - A list of heuristics that can automate labeling
  - Typically provided by subject matter experts
- Noisy labels have a certain probability of being correct, not 100%
- Objective: learn a generative model to determine weights for weak supervision sources

# Snorkel

- Project started at Stanford in 2016
- Programmatically building and managing training datasets without manual labeling
- Automatically: models, cleans, and integrates the resulting training data
- Applies novel, theoretically-grounded techniques
- Also offers data augmentation and slicing

# Data programming pipeline in Snorkel



Users write labeling functions to generate noisy labels for unlabeled data

A generative model is used to de-noise and weight the labels

The labels are used to train a model

# Snorkel labeling functions

```python
from snorkel.labeling import labeling_function


@labeling_function()
def lf_keyword_my(x):
    """Many spam comments talk about 'my channel', 'my video', etc."""
    return SPAM if "my" in x.text.lower() else ABSTAIN


@labeling_function()
def lf_short_comment(x):
    """Non-spam comments are often short, such as 'cool video!'."""
    return NOT_SPAM if len(x.text.split()) < 5 else ABSTAIN
```

# Key points

- Semi-supervised learning:
    - Applies the best of supervised and unsupervised approaches
    - Using a small amount of labeled data boosts model accuracy
- Active learning:
    - Selects the most important examples to label
    - Improves predictive accuracy
- Weak supervision:
    - Uses heuristics to apply noisy labels to unlabeled examples
    - Snorkel is handy framework for weak supervision

DeepLearning.AI

# Data Augmentation

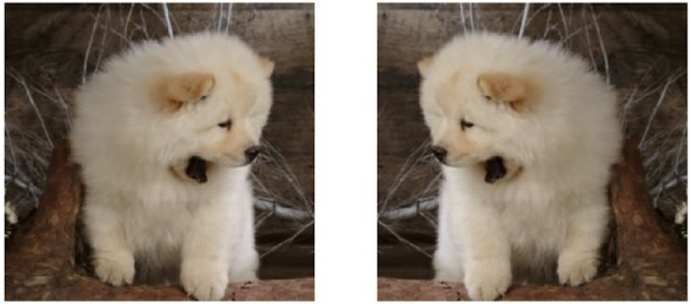
## Data Augmentation

# Outline

- Generating synthetic data
- Augmenting an image dataset: CIFAR-10 example
- Other advanced techniques

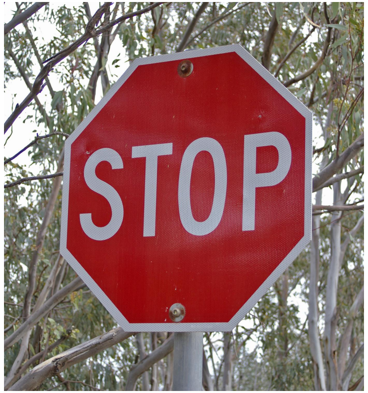
# How do you get more data?

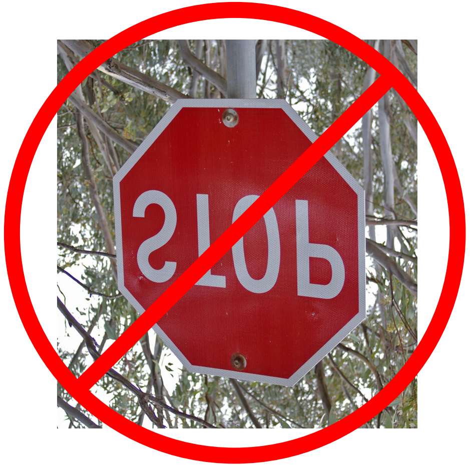
- Augmentation as a way to expand datasets
- One way is introducing minor alterations
- For images: flips, rotations, etc.



# How does augmentation data help?

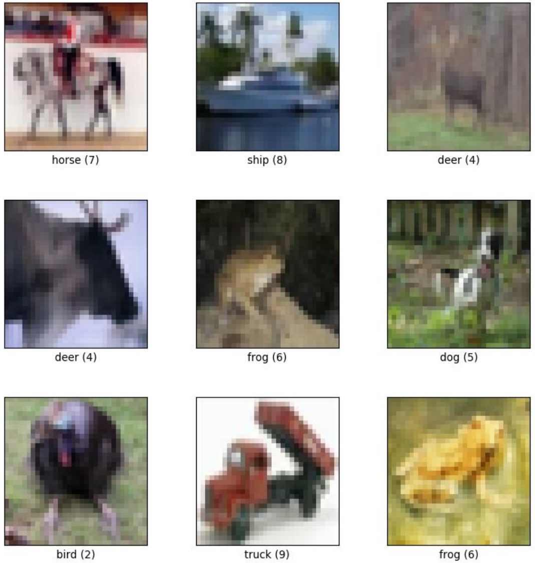
- Adds examples that are similar to real examples
- Improves coverage of feature space
- Beware of invalid augmentations!





# An example: CIFAR-10 data set

- 60,000 32x32 color images
- 10 classes of objects
  (6,000 images per class)



# Data augmentation on CIFAR-10

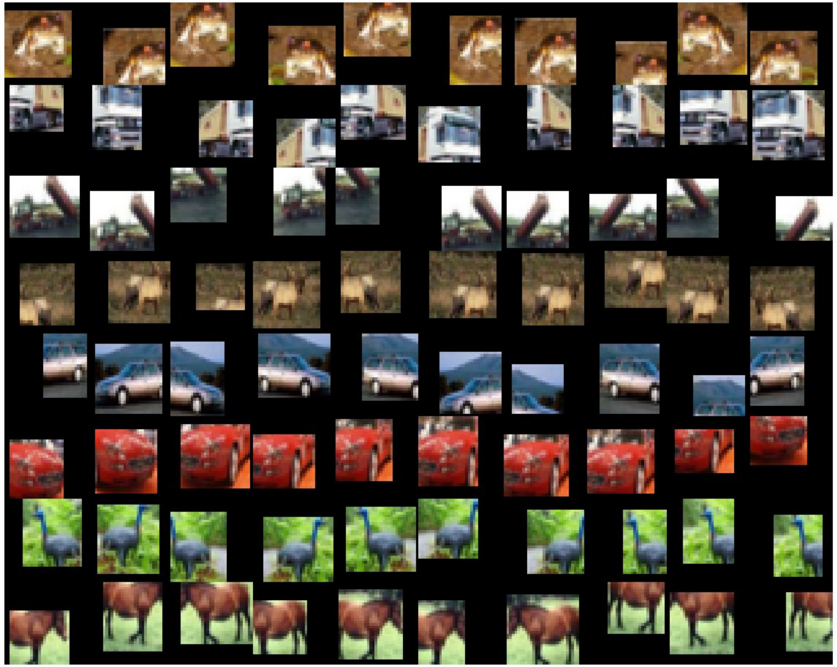
Let's augment the CIFAR-10 dataset with the following steps:

1. Pad the image with a black, four-pixel border
2. Randomly crop a 32 x 32 region from the padded image
3. Flip a coin to determine if the image should be flipped horizontally left/right

# Defining the augment operation

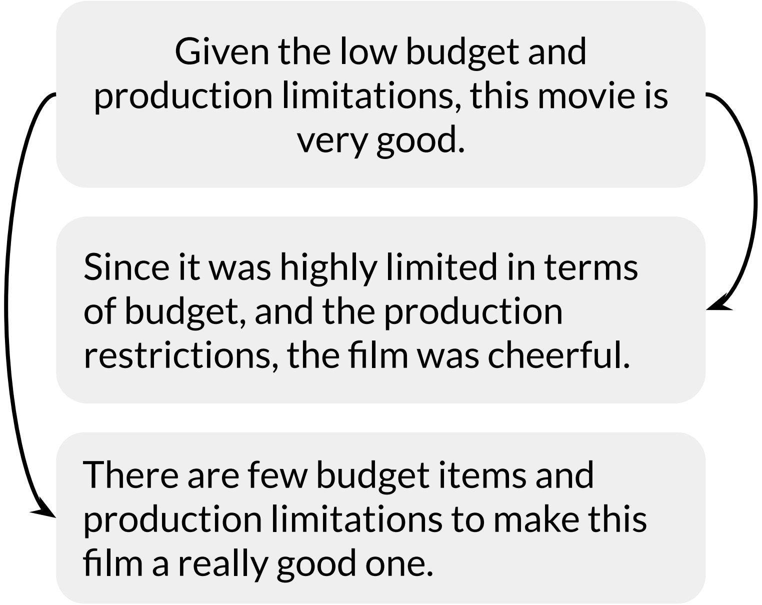
```python
def augment(x, height, width, num_channels):
    x = tf.image.resize_with_crop_or_pad(x, height + 8, width + 8)
    x = tf.image.random_crop(x, [height, width, num_channels])
    x = tf.image.random_flip_left_right(x)
    return x
```

# Augmented examples



# Other Advanced techniques

- Semi-supervised data augmentation e.g., UDA, semi-supervised learning with GANs
- Policy-based data augmentation e.g., AutoAugment



Given the low budget and production limitations, this movie is very good.

Since it was highly limited in terms of budget, and the production restrictions, the film was cheerful.

There are few budget items and production limitations to make this film a really good one.

# Key points on data augmentation

- It generates artificial data by creating new examples which are variants of the original data
- It increases the diversity and number of examples in the training data
- Provides means to improves accuracy, generalization, and avoiding overfitting

DeepLearning.AI

# Preprocessing More Data Types

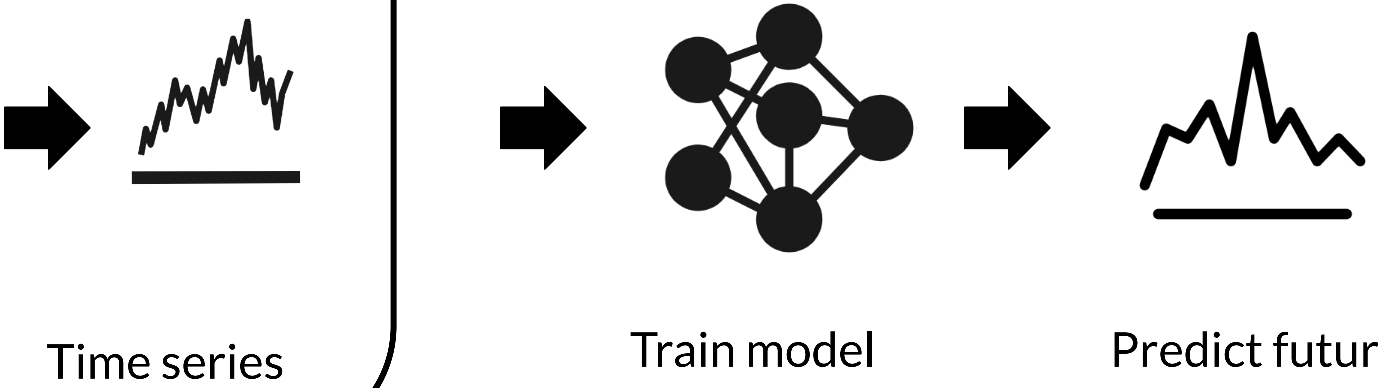
## Time series

# A note on different types of data

- TFX pre-processing capabilities for multiple data types:
    - Images
    - Video
    - Text
    - Audio
    - Time series
- Optional notebook on images
- Two optional notebooks on time series

# Time series data

| time | y(t) |
|---|---|
| 0 | 0.27 |
| 1 | 0.33 |
| 2 | 0.48 |
| 3 | 0.67 |
| ⋮ | ⋮ |
| t-1 | 0.2 |
| t | ??? |



Time series

Train model

Predict future

*“It is difficult to make predictions, especially about the future.”*

- Karl Kristian Steincke

# Time series forecasting

- Time Series forecasting predicts future events by analyzing data from the past
- Time series forecasting makes predictions on data indexed by time
- Example:
    - Predict future temperature at a given location based on historical meteorological data

# Time series dataset: Weather prediction

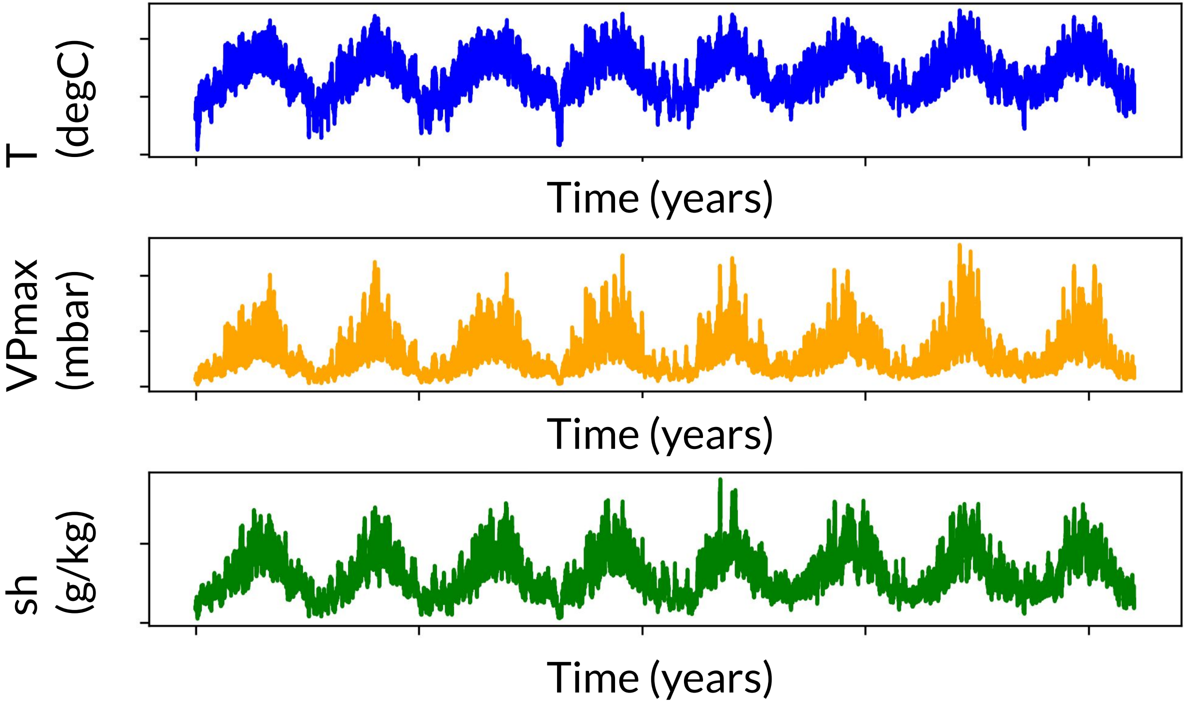
We will use the weather time series dataset recorded by the Max Planck Institute for Biogeochemistry:

- to preprocess time series data with TensorFlow Transform
- to convert data into sequences of time steps:
    - Making data ready to train a long short-term memory (LSTM) recurrent neural network (RNN)

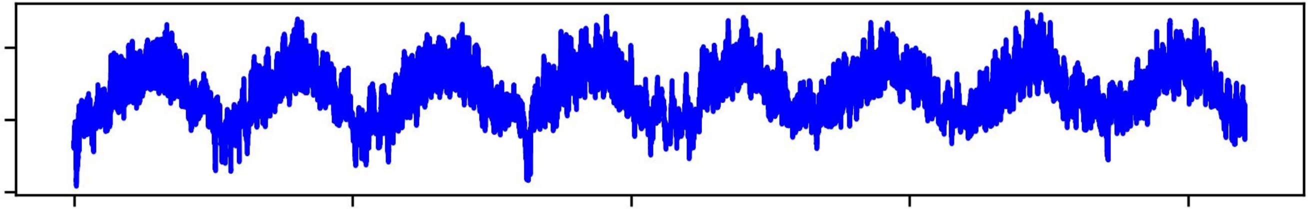
# Weather time series dataset

- There are 14 variables:
  - P(mbar), T (degC), Tdew (degC), rh (%), VPmax (mbar), VPact (mbar), VPdef (mbar), sh (g/kg), H20C (mmol/mol), rho (g/m**3), wv (m/s), max.xv (m/s), wd (deg)
  - Target is  T (degC)
- Observations recorded every 10 minutes
  - 6 observations per hour
  - 144 (6 X 24) observations in a day.

# Data visualizations



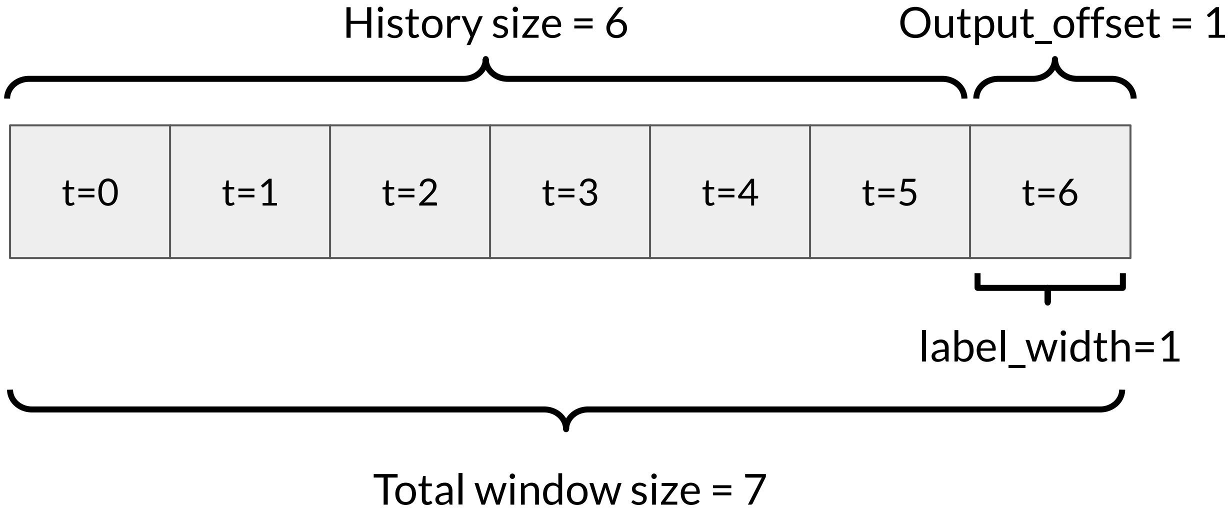
# Windowing data



- Windowing makes sense.
- tf.data.Datasets.window() converts times series data to depend on past observations.

# Windowing strategies in single step time series

A model that makes a prediction 1h into the future, given 6h of history would need a window like this:



History size = 6

Output_offset = 1

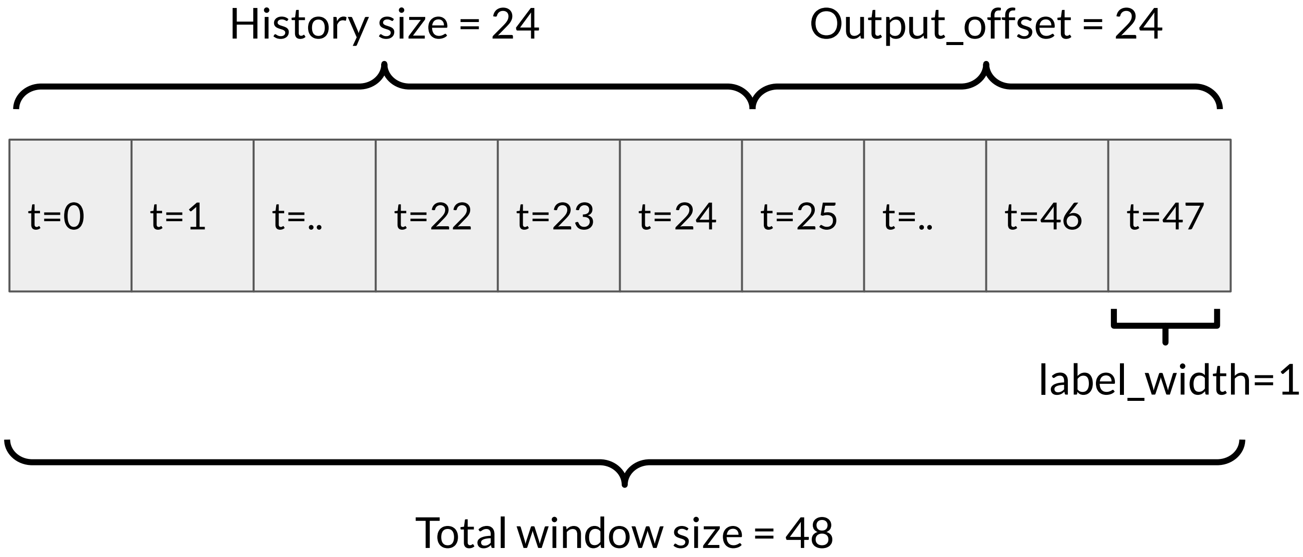
| t=0 | t=1 | t=2 | t=3 | t=4 | t=5 | t=6 |
|---|---|---|---|---|---|---|

label_width=1

Total window size = 7

# Windowing strategies in single step time series

Predict next 24 hours given 24 hours of history



History size = 24

Output_offset = 24

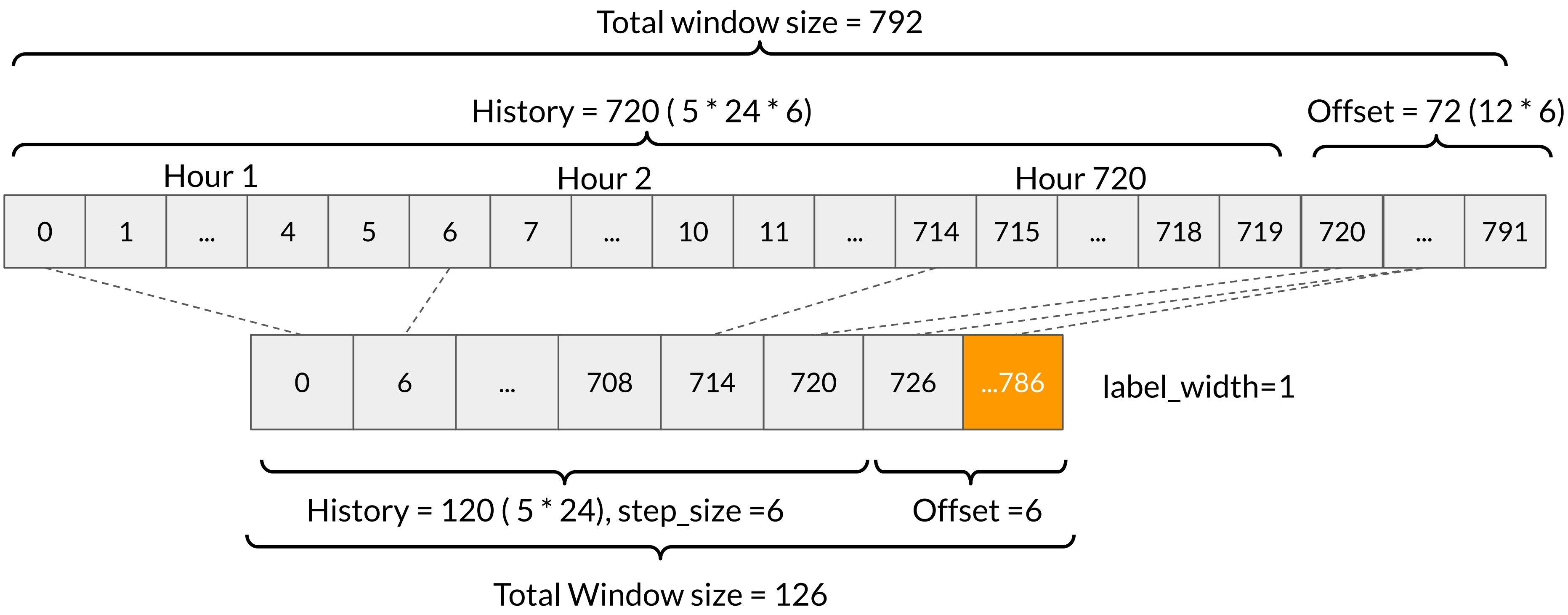
| t=0 | t=1 | t=.. | t=22 | t=23 | t=24 | t=25 | t=.. | t=46 | t=47 |
|---|---|---|---|---|---|---|---|---|---|

label_width=1

Total window size = 48

# Windowing strategy for the problem

Sample one observation every hour with step size = 6



Total window size = 792

History = 720 ( 5 * 24 * 6)

Offset = 72 (12 * 6)

Hour 1

Hour 2

Hour 720

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | ... | 4 | 5 | 6 | 7 | ... | 10 | 11 | ... | 714 | 715 | ... | 718 | 719 | 720 | ... | 791 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 6 | ... | 708 | 714 | 720 | 726 | ...786 |

label_width=1

History = 120 ( 5 * 24), step_size =6

Offset =6

Total Window size = 126

# Optional notebook: what will you do?

- Data processing with TFX to extract features
- Segment data into windows
- Save data in TFRecord format
- Make it ready for training an LSTM model

DeepLearning.AI

# Preprocessing More Data Types

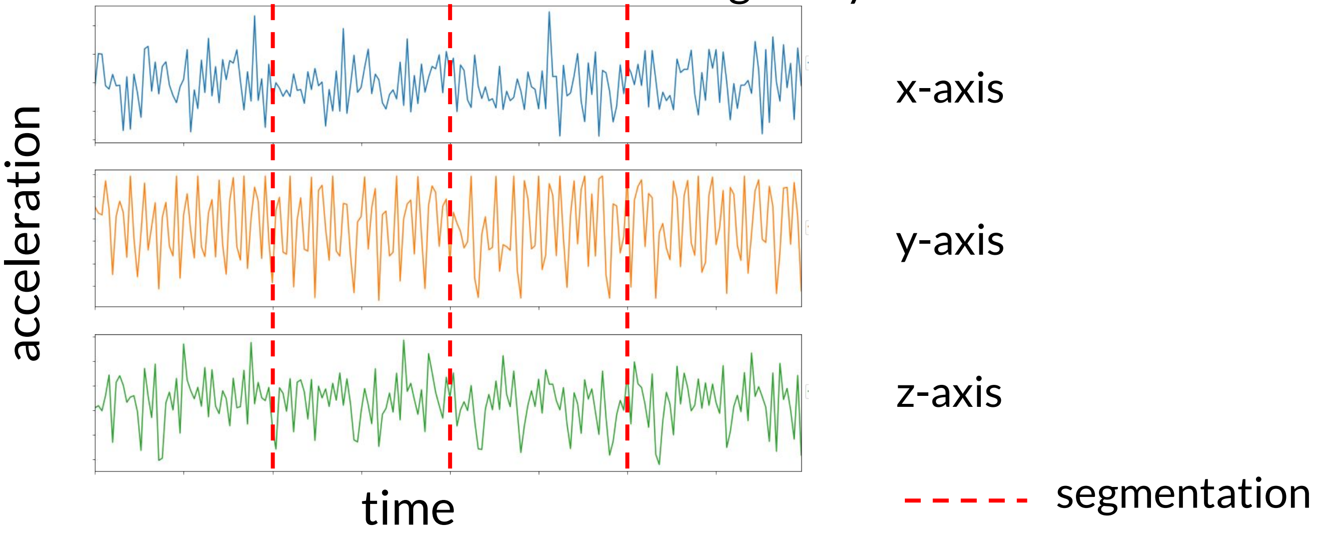
## Sensors and Signals

# Sensors and Signals

- Signals are sequences of data collected from real time sensors
- Each data point is indexed by a timestamp
- Sensors and signals data is thus time series data
- Example: classify sequences of accelerometer data recorded by the sensors on smartphones to identify the associate activity

# Human activity recognition (HAR)

- HAR tasks require segmentation operations
  - Raw inertial data from wearables fluctuate greatly over time



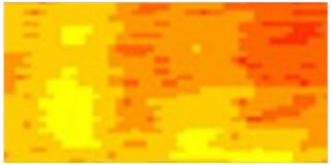
# Human activity recognition (HAR)

- Segmented data should be transformed for modeling
- Different methods of transformation:
    - Spectrograms (commonly used)
    - Normalization and encoding
    - Multichannel
    - Fourier transform

Raw Data

Multichannel Data

Spectogram



# Optional notebook: what will you do?

- Work with Human Activity Recognition Dataset (WISDM):
  - Preprocess with TensorFlow Transform
  - Use tf.data.Datasets.window() for converting times series data to depend on past observations

DeepLearning.AI

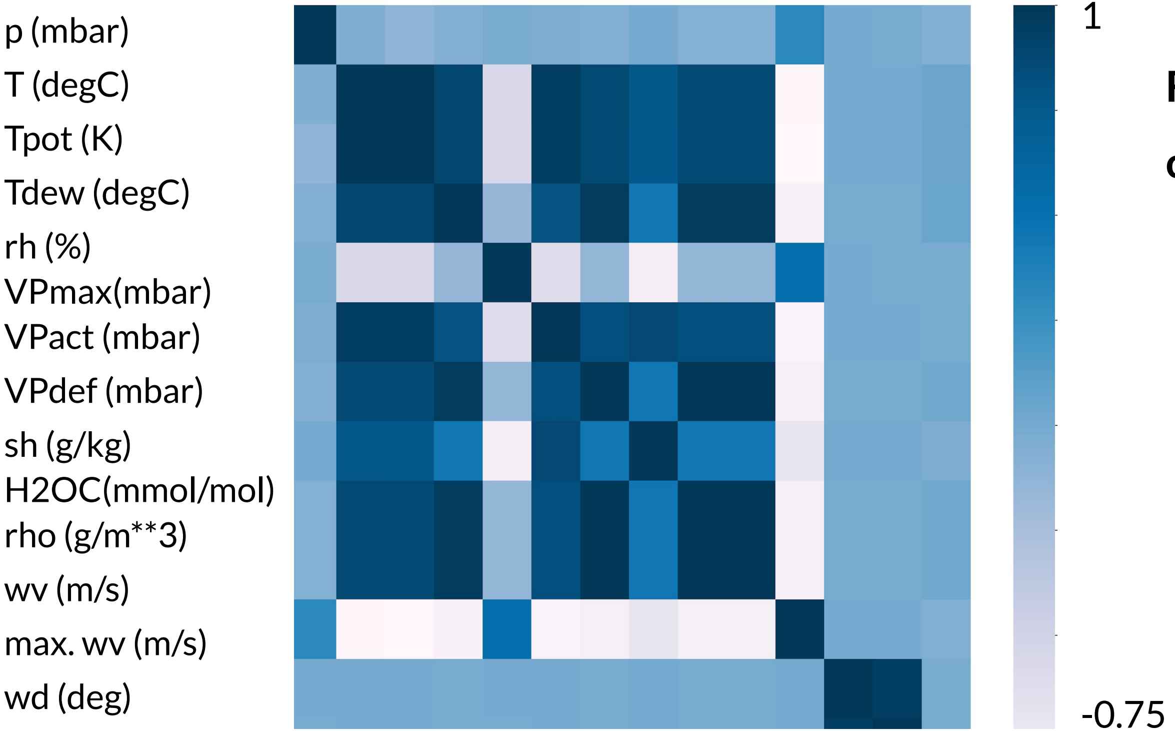
Preprocessing More Data Types

# Time Series Assignment Walkthrough

# Detecting inconsistent data

| | count | mean | std | **min** | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| wv (m/s) | 420551 | 1.702 | 65.44 | **-9999.0** | 0.99 | 1.76 | 2.86 | 28.49 |
| max. wv (m/s) | 420551 | 3.06 | 69.01 | **-9999.0** | 1.76 | 2.96 | 4.74 | 23.5 |

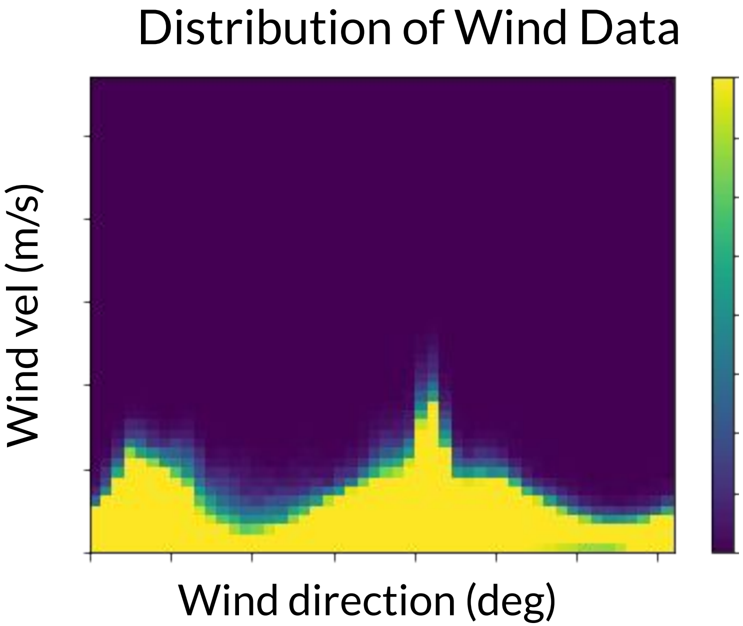
# Feature correlation



**Features to Remove due to high correlations:**

1. Tpot (k)
2. Tdew (degC)
3. VPact
4. H2OC
5. max. wv

# Consistent wind direction and velocity



- Wind direction is in units of degrees
- $360^0$ and $0^0$ should be close to each other and wrap around smoothly
- Wind direction doesn't matter if wind is not blowing
- Using TF Transform we will convert wind direction and wind velocity into wind vector

# Preprocessing the date time feature

- Date Time columns is in string format
- Weather data has clear daily and yearly periodicity

Transformation Needed:

- Convert string date time to timestamp
- Use sin and cos to convert it into 2 features:
  - Time of the day
  - Time of the year

# Reading and cleaning input data

- Read the Input data using beam.io.ReadFromText()
- Clean data using a Beam Transform:
    - Decode the input lines to transform into feature value pairs using a schema
    - Remove extreme min values of -9999.0 from wind velocity and max wind velocity features
    - Convert Date Time feature to timestamp

# Train test splits

- First 300,000 records are used for training, the remaining for testing
- You will partition the dataset using the beam.Partition transform
- beam.Partition needs a partition function which defines logic of partition

# Preprocessing the dataset

- Delete unwanted features (Feature Selection)
- Transform Wind Direction and Wind Velocity to a wind vector
- Transform timestamp DateTime to ‘Time of Year’ and ‘Time of Day’
- Normalize float features

# Advanced Labeling, Augmentation, and Preprocessing

Semi-supervised labeling

- Graph-based approach
- Active learning

Weak supervision

- Snorkel

Data augmentation

- Image transformations
- Policy-based

Time series

- Windowing
- Sensors and signals