Tips and Tricks

Original Slides by:
Courtney Anne Ngo
Daniel Stanley Tan, PhD
Arren Antiquioa

Updated (AY 2024 – 2025 T1) by:
Thomas James Tiam-Lee, PhD

DE LA SALLE UNIVERSITY
RELIGIO MORES CULTURA
MANILA

College of
Computer Studies

About this Slide

- These slides cover some general tips and tricks that one can apply when in practice when training machine learning models.

  - Preprocessing techniques
  - Feature and label encoding techniques
  - Data augmentation
  - Babysitting the model

Preprocessing

- Things to consider before training the model:

  - Data cleaning
  - Exploratory data analysis
  - Scaling and normalization

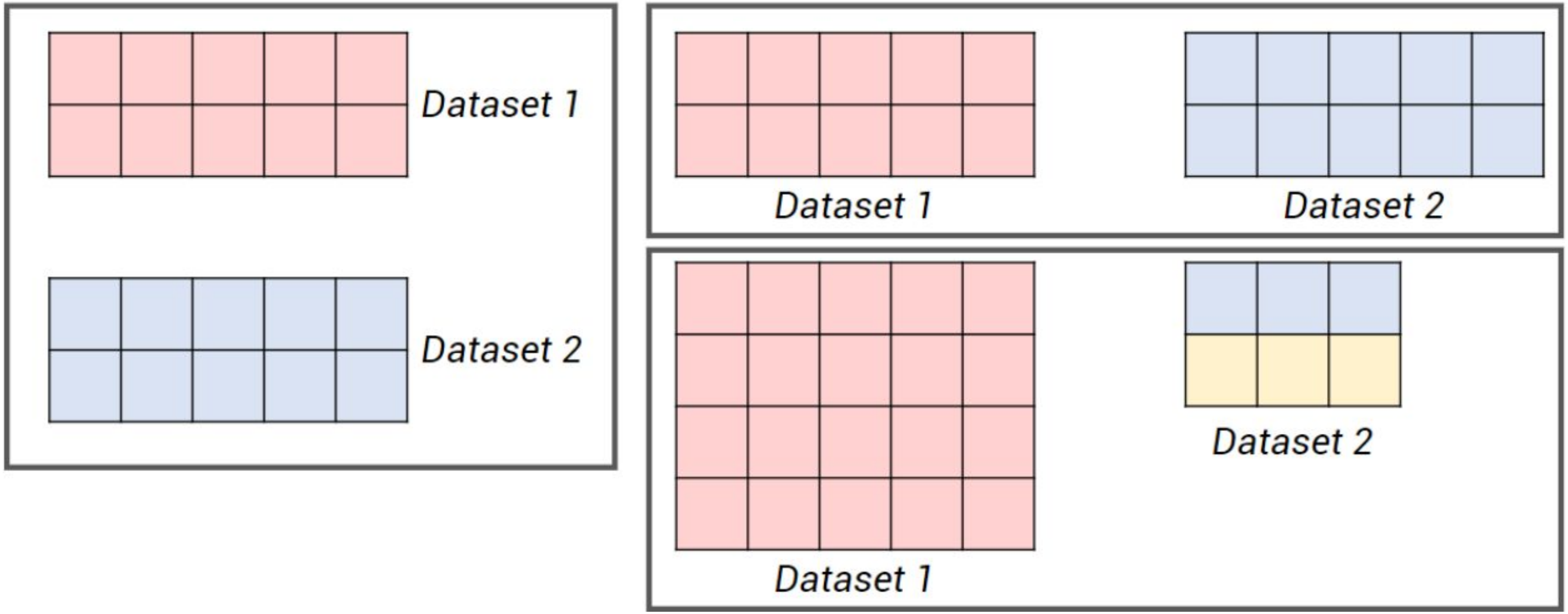
Data Cleaning

- Handling missing values
  - Dropping
    - Remove rows and/or columns that contain missing values
  - Imputation
    - Use the average value
    - Use the average value of most similar instance/s

Age	Height	Weight
25	175	70
32	163	65
48	180	85
23	?	68
62	158	72
38	188	92
24	165	68

Data Cleaning

- Data coming from multiple sources



Data Cleaning

- Different representations of text / numbers

Sex
M
Female
F
Male
m

Sex
M
F
F
M
M

Age
18
20
'19'
20
19

Age
18
20
19
20
19

Data Cleaning

- Duplicate data
  - In some case, it makes sense for the data to contain duplicates.
  - In some cases, duplicate data is caused by an error in encoding

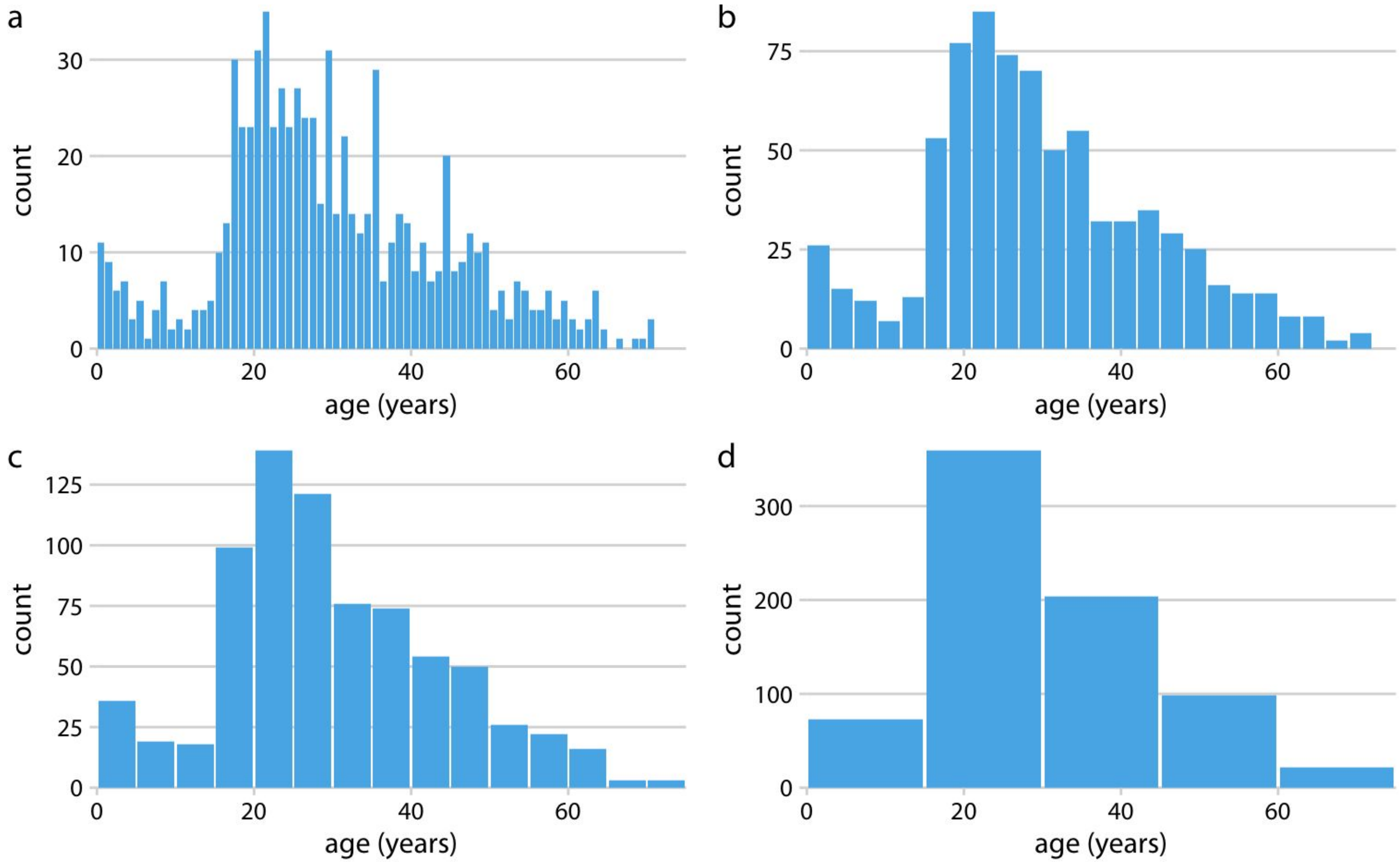
Device ID	Latitude	Longitude
A1264	14.63	121.03
C4286	14.61	120.99
G7921	14.67	121.05
C4286	14.61	120.99
A1773	14.51	120.41
G1279	14.59	121.90
C4286	14.61	120.99

Exploratory Data Analysis

- EDA must cover:
  - Different information available in the data
  - Range of values of each variable
  - Distributions of each variable
  - Presence of outliers, if any
  - Correlations within the different variables

Exploratory Data Analysis

- Distribution of variables



Exploratory Data Analysis

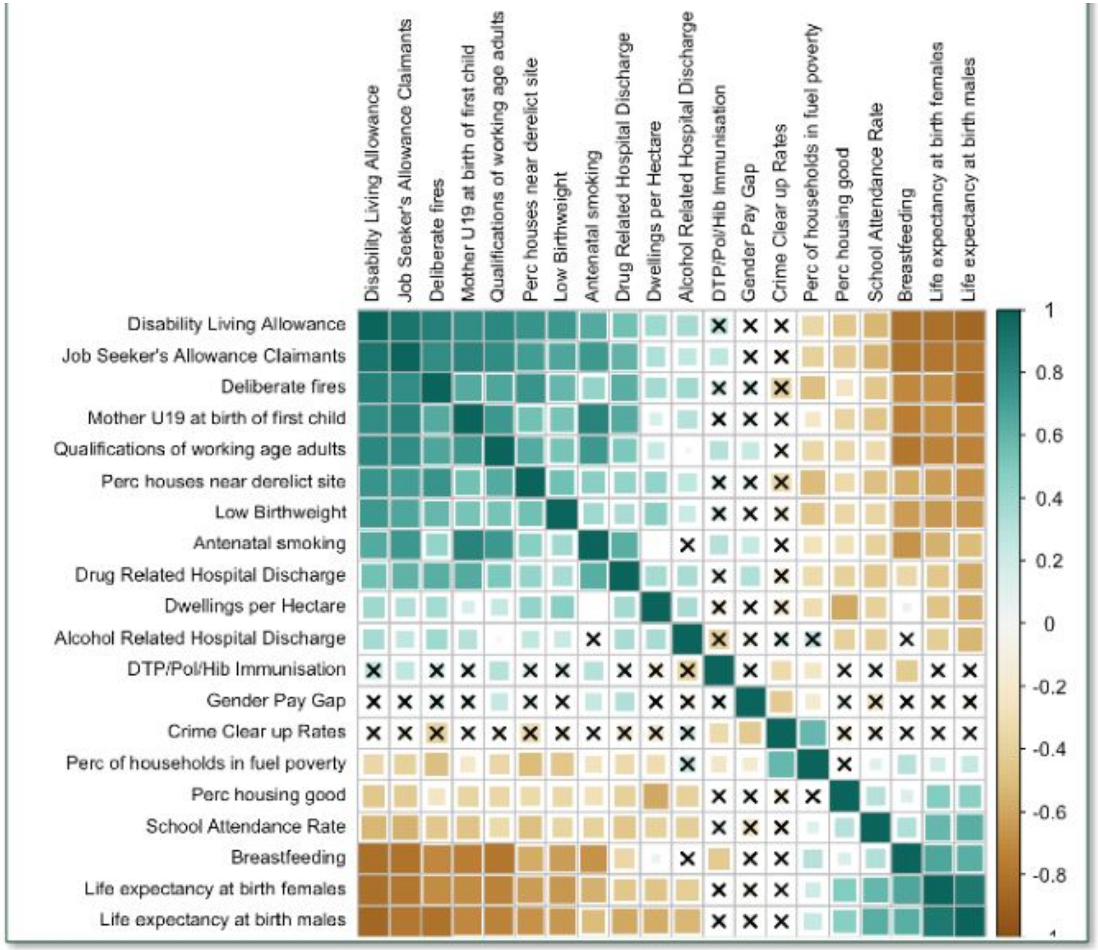
- Handling outliers
  - Remove
  - Transform (logarithmic, square root, etc.)
  - Impute (replace)

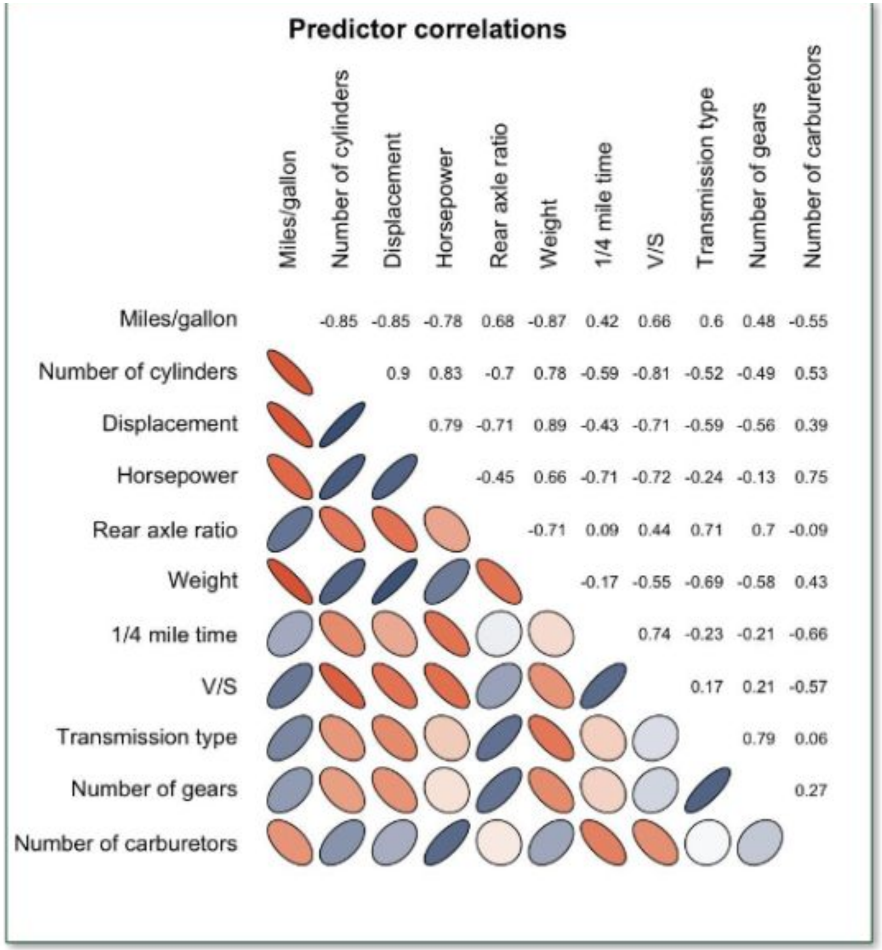
- Note: the choice for handling outliers depends on the context of the data and the problem. You must make sure that the handling of outliers does not substantially affect the integrity of the training data itself!

Exploratory Data Analysis

- Correlation matrix

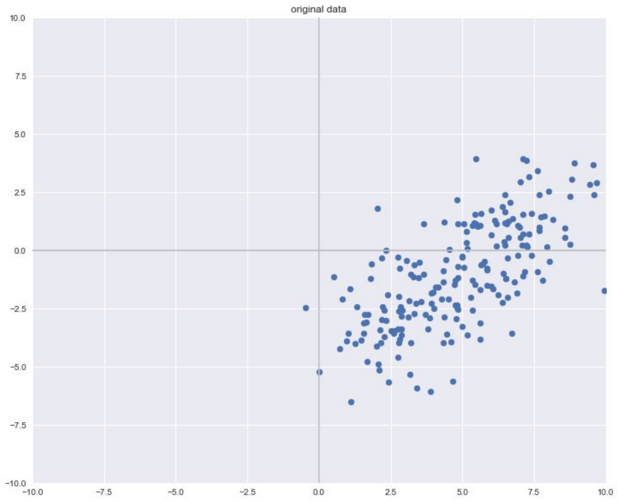
Note: Some ML models do not work very well when the features are highly correlated!



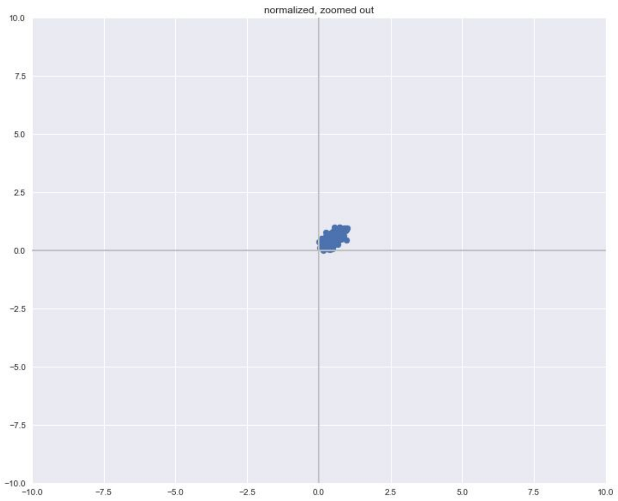


Normalization

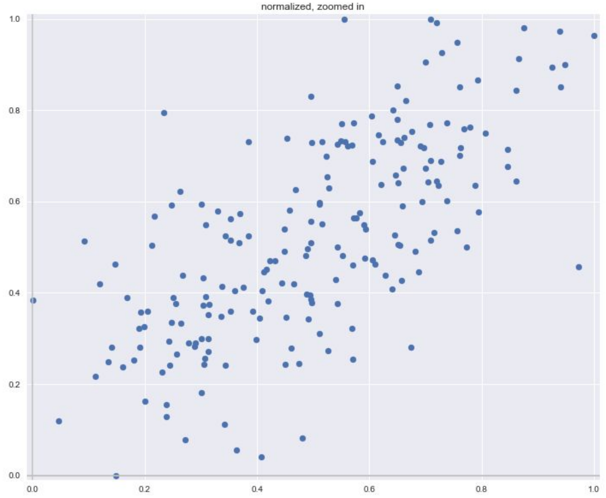
$$x_{normalized} = \frac{x - \min(x)}{\max(x) - \min(x)}$$



Original Data



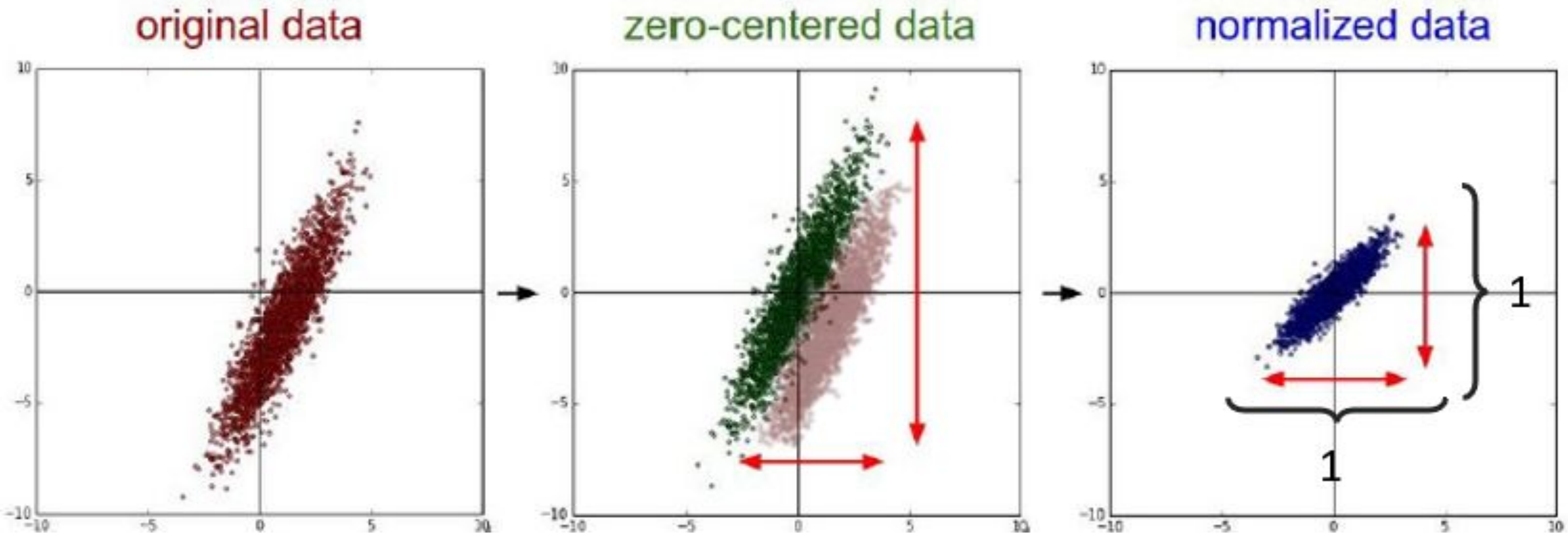
Normalized



Normalized (zoomed in)

Standardization

$$x_{standardized} = \frac{x - mean(x)}{stddev(x)}$$



Normalization Vs. Standardization

- **Normalization**
  - Values fall between 0 and 1
  - More sensitive to outliers
  - Useful when we don’t know the distribution of the underlying data
- **Standardization**
  - Values are not constrained to a range
  - Less sensitive to outliers
  - Useful when the data follows a normal or Gaussian distribution

Normalization / Standardization Pipeline

- When normalizing, the **min** and **max** should only be based on the training data (not the test data).

- When standardizing, the **mean** and **stddev** should only be based on the training data (not the test data).

- The test data should then be normalized / standardized according to the metrics of the training data **to avoid data leakage**.

Feature Selection

- Identifying the relevant features that are helpful for prediction

- Common ways to identify features:
  - Domain knowledge
  - Statistical methods
    - Correlation
    - Chi-square test
    - ANOVA

Feature and Label Encoding

- Some machine learning models require you to use encodings in representing features / labels.
- **sklearn** has convenient functions that allow you perform these encodings.

Label Encoding

- Assign a unique integer for each class

	size	color	type
0	medium	red	rayon
1	large	green	polyester
2	small	blue	cotton
3	medium	white	cotton
4	extra large	gray	cotton
5	large	black	polyester
6	medium	green	rayon
7	extra small	blue	linen
8	medium	grey	cotton
9	large	green	polyester

	size	color	type
0	medium	red	3
1	large	green	2
2	small	blue	0
3	medium	white	0
4	extra large	gray	0
5	large	black	2
6	medium	green	3
7	extra small	blue	1
8	medium	grey	0
9	large	green	2

One-Hot Encoding

- Makes a new feature for each class.

	size	color	type
0	medium	red	rayon
1	large	green	polyester
2	small	blue	cotton
3	medium	white	cotton
4	extra large	gray	cotton
5	large	black	polyester
6	medium	green	rayon
7	extra small	blue	linen
8	medium	grey	cotton
9	large	green	polyester

```
array([[0., 0., 0., 1.],
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.],
       [0., 0., 1., 0.]])
```

Ordinal Encoding

- Similar to label encoding but there is an ordinal value

	size	color	type
0	medium	red	rayon
1	large	green	polyester
2	small	blue	cotton
3	medium	white	cotton
4	extra large	gray	cotton
5	large	black	polyester
6	medium	green	rayon
7	extra small	blue	linen
8	medium	grey	cotton
9	large	green	polyester

```
0    2
1    3
2    1
3    2
4    4
5    3
6    2
7    0
8    2
9    3
Name: size, dtype: int64
```

```
clothes_sizes_dict= {
    "extra small" : 0,
    "small" : 1,
    "medium" : 2,
    "large" : 3,
    "extra large" : 4
}
```

Techniques for Improving Model Performance

- Data Augmentation
- Diagnosing the ML Model
- Error Analysis
- Hyperparameter Tuning

Data Augmentation

- In most cases, ML models will improve performance with **more data**!
- Problem: more data is often expensive, and in some cases, impossible to obtain
- Solution: generate artificial data from existing ones

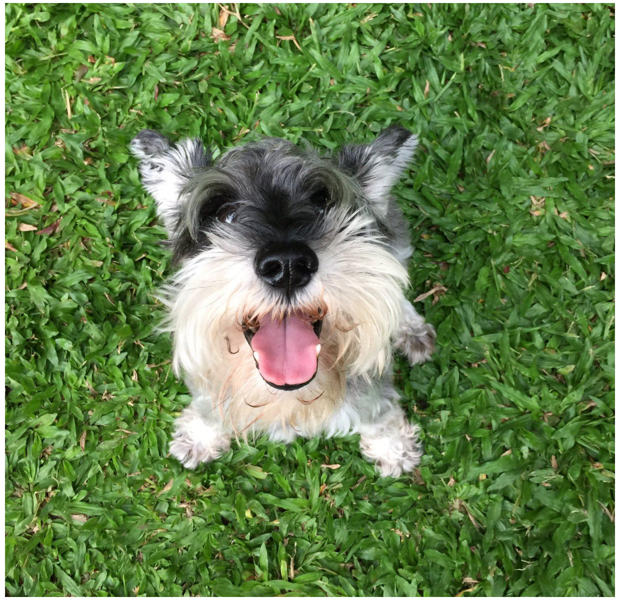
Data Augmentation Example (CV)

- Augmenting the image with different transformations will help enlarge your dataset.

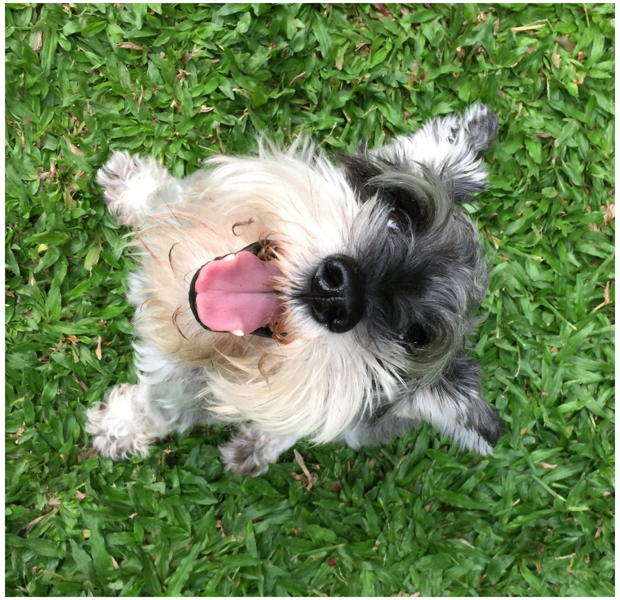
- **Plus, it also helps train the model to learn more situations that it should recognize!**

Data Augmentation Example (CV)

- CNN models[1] are inherently spatially invariant.
- Data augmentation can help solve this!



CNN : dog



CNN : ???

[1] Convolutional neural networks, a kind of neural network usually used for computer vision

Data Augmentation Example (CV)

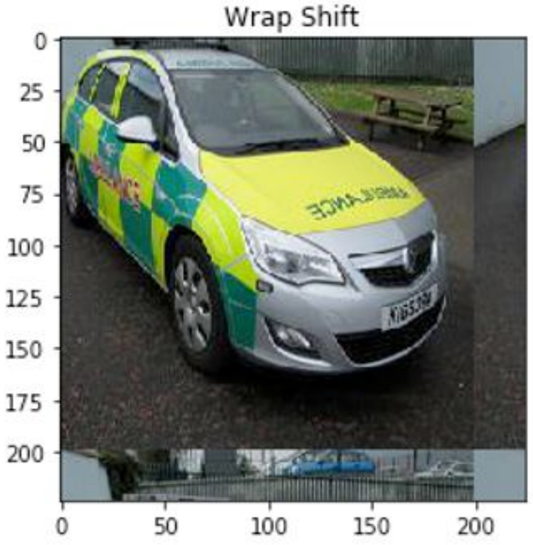
- Examples of data augmentation techniques in CV

Rotate

- In varying angles

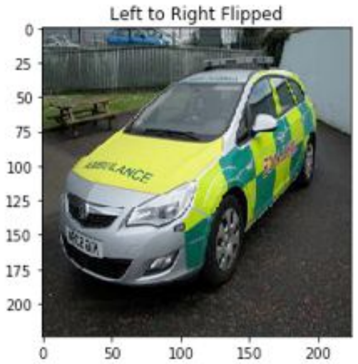
Shift

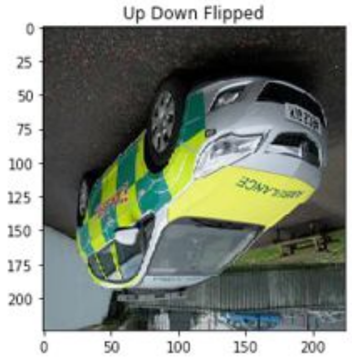
- In varying pixels (left and top)



Flip

- to r, u to d



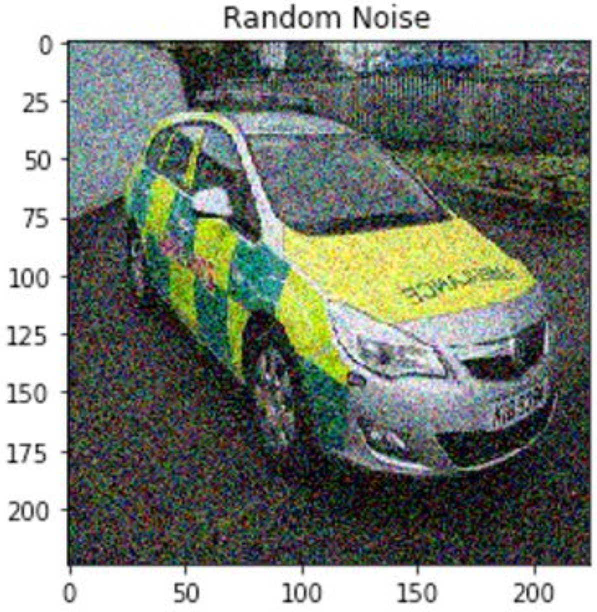


Data Augmentation Example (CV)

- Examples of data augmentation techniques in CV

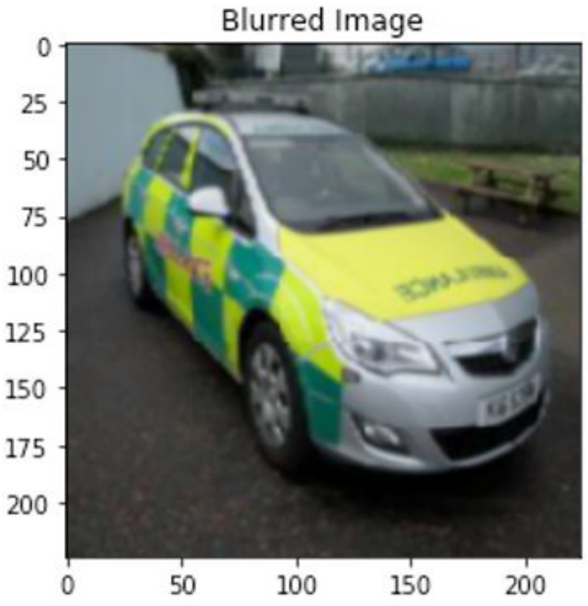
Add noise

- In varying degrees



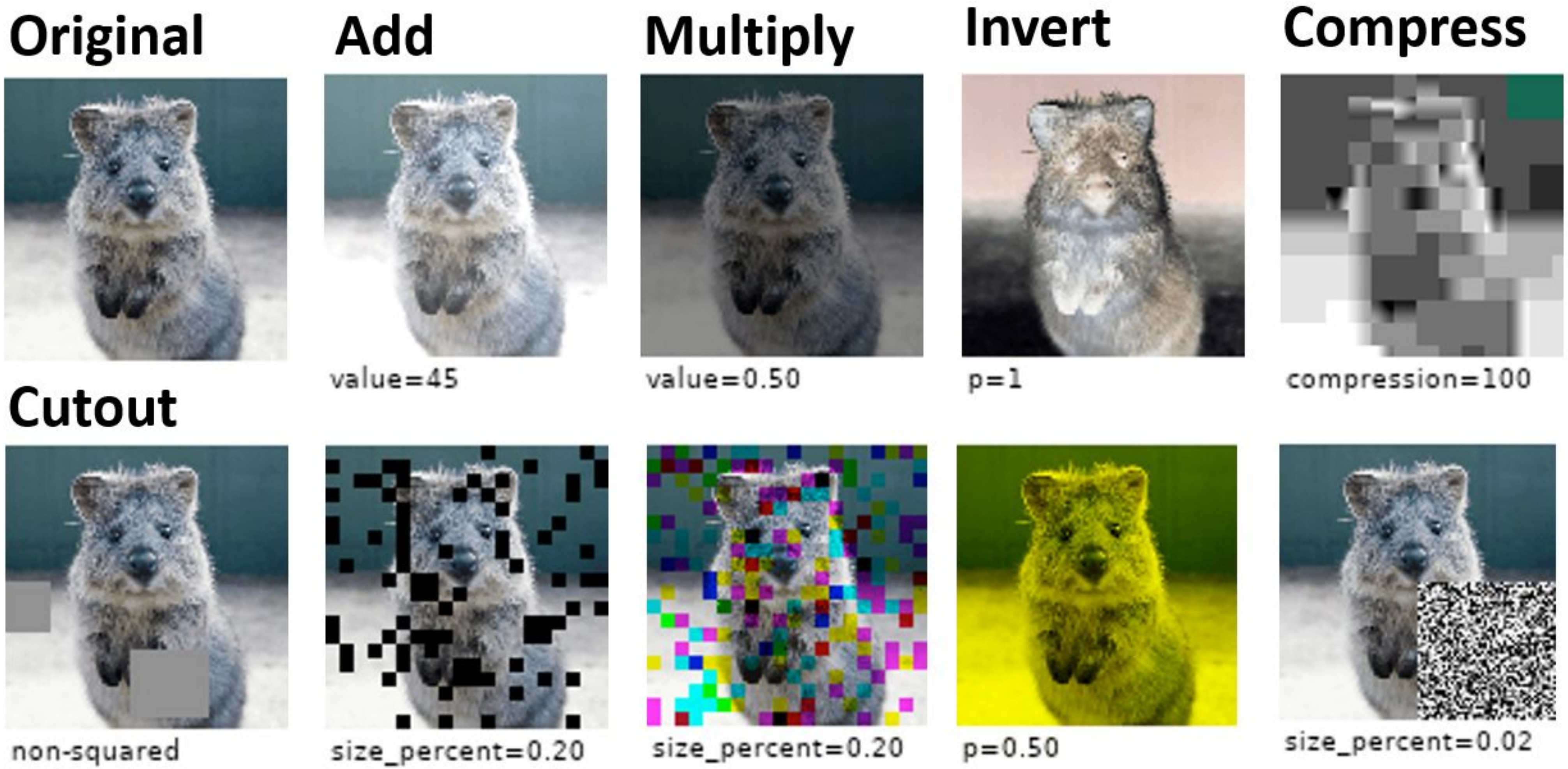
Blur

- In varying sigma



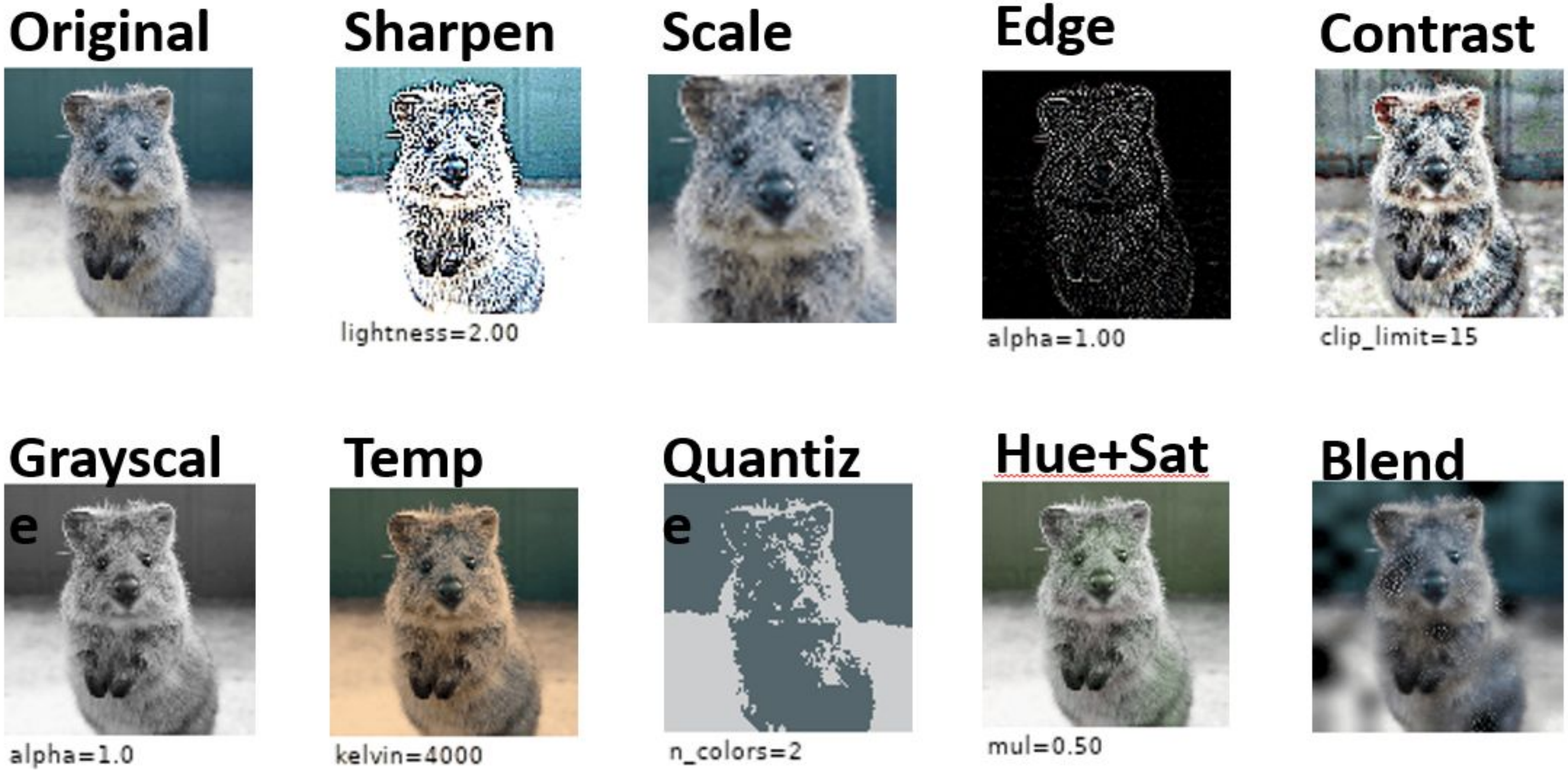
Data Augmentation Example (CV)

- Examples of data augmentation techniques in CV



Data Augmentation Example (CV)

- Examples of data augmentation techniques in CV



Data Augmentation Example (CV)

- Examples of data augmentation techniques in CV

weather

FastSnowyLandscape (lightness_multiplier=2.0)	Clouds	Fog	Snowflakes	Rain
 lightness_threshold=50				
See also: CloudLayer, SnowflakesLayer, RainLayer				

Data Augmentation Example (NLP)

- **Task:** Optical Character Recognition (OCR)
  - From original data, replace characters with similar looking fonts

Data Augmentation Example (NLP)

- **Task: Text understanding tasks**
  - From original data, replace random characters with likely mistyped characters

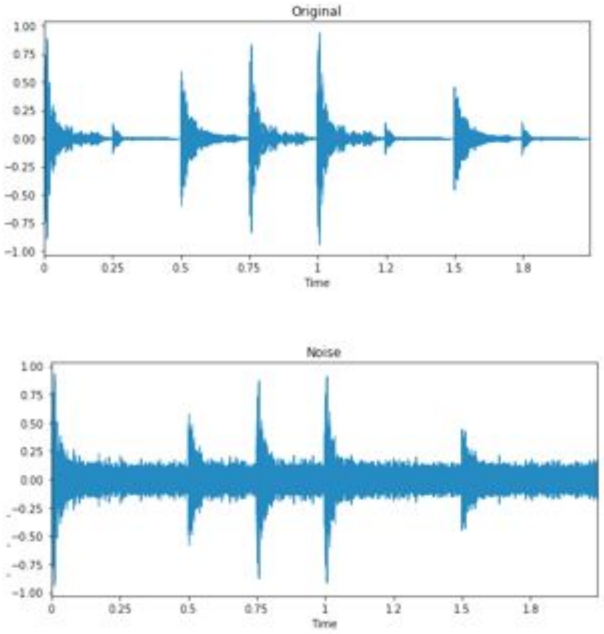
```
The quick brown
fox jumps.
```

➡

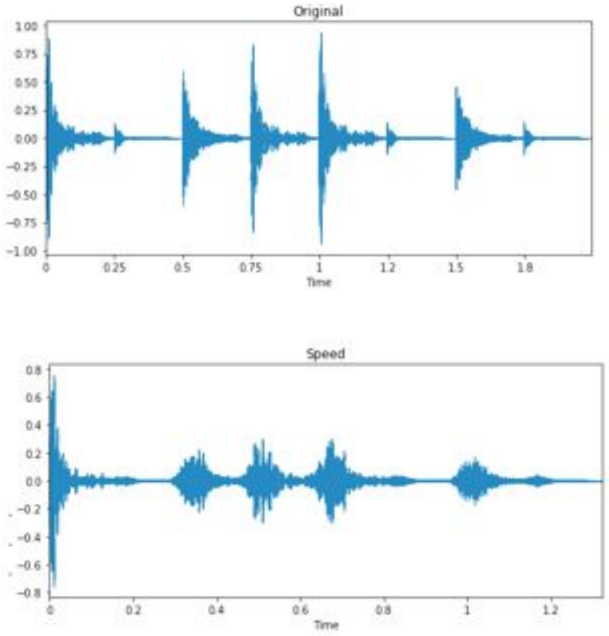
```
The quock brown
fox jumps.

The quick brown
foc jumps.

The qick brown
fox junp.
```

Data Augmentation Example (Audio)

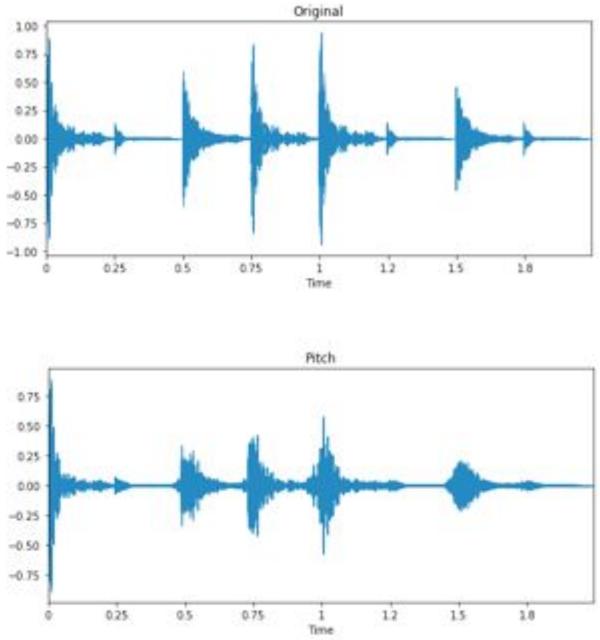
Add noise



Speed up/down



Change pitch



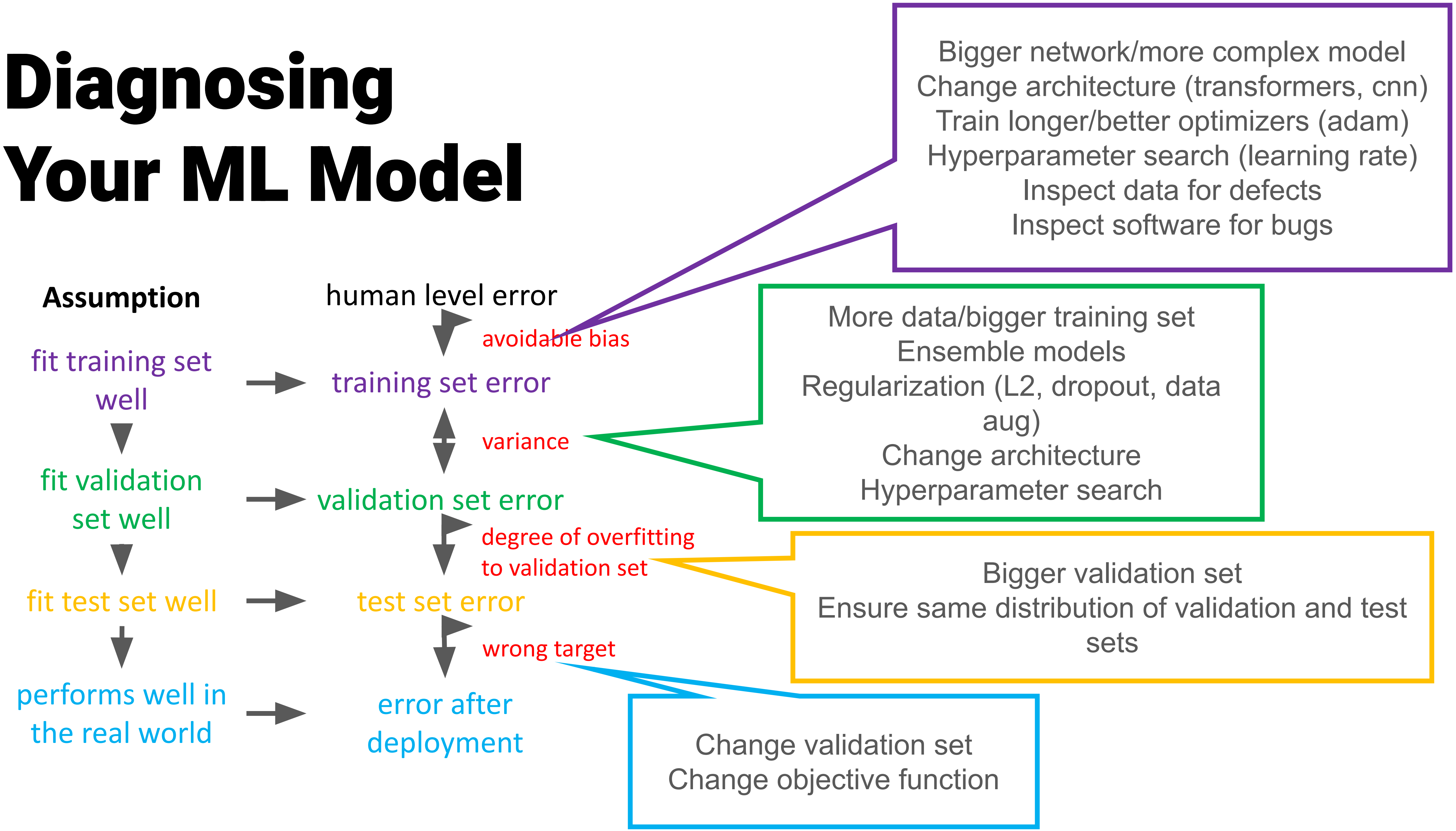
Reminders

- **Never use augmented data on the validation / testing set.**

- Validation / testing sets are meant to represent data in the real-world, since we want to use it as a measure of how well our model performs in the wild. Therefore, it should **not** contain any artificial data.

- When using cross-validation, make sure data augmentation is **only applied to the training set** in every fold.

Diagnosing Your ML Model



Human Level Error

- **Check what the model is mis-classifying.**

- Can an expert human even classify this correctly? If not, we can't expect that a computer can.

- In this example, the generation of the training data may have been faulty.

Image:

Label: 26624

Bias-Variance Analysis

Case 1

human error	1%
training error	8%
validation error	10%

7% avoidable bias

High avoidable bias, model is underfitting

Case 2

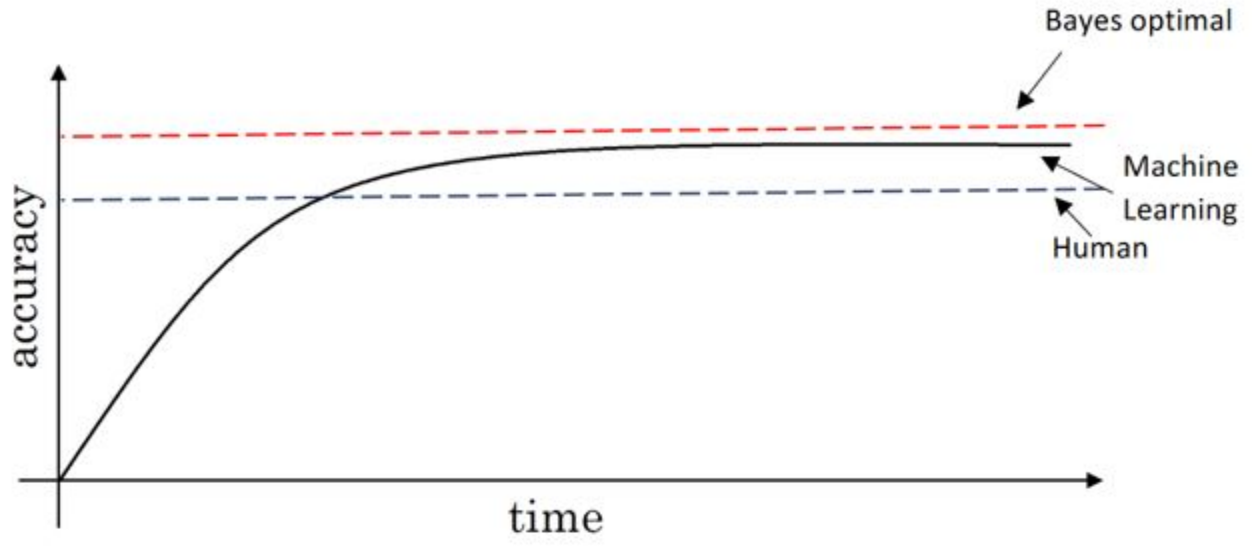
human error	7.5%
training error	8%
validation error	16%

0.5% avoidable bias

8% gap

Low avoidable bias, model is complex enough to fit data

Big gap between train and validation error is sign of overfitting



We should use the Bayes optimal error in theory

Bayes optimal error == lowest possible error

Same with irreducible error

Use human error if Bayes optimal error is not available

Bias-Variance Analysis

Task: Medical, classification

	error
typical human	3.0%
typical doctor	1.0%
experienced doctor	0.7%
team of experienced doctors	0.5%

Since our task is medical, we should compare our model to a doctor's level

	error	
	case 1	case 2
human error (Bayes optimal error proxy)	1.0%	
	0.7%	
	0.5%	
training error	5.0%	1.0%
val error	6.0%	5.0%

case 1

train – human (error)	4 - 4.5%
val – train (error)	1.0%

Avoidable bias is big, focus on reducing bias

case 2

train – human (error)	0 - 0.5%
val – train (error)	4.0%

Variance is big, focus on reducing variance

Train-Val-Test Split

- We have climate data for different countries

- **Idea 1:** Split the data like this:

- Is this a good way to split the data?

Train-Val-Test Split

- We have climate data for different countries

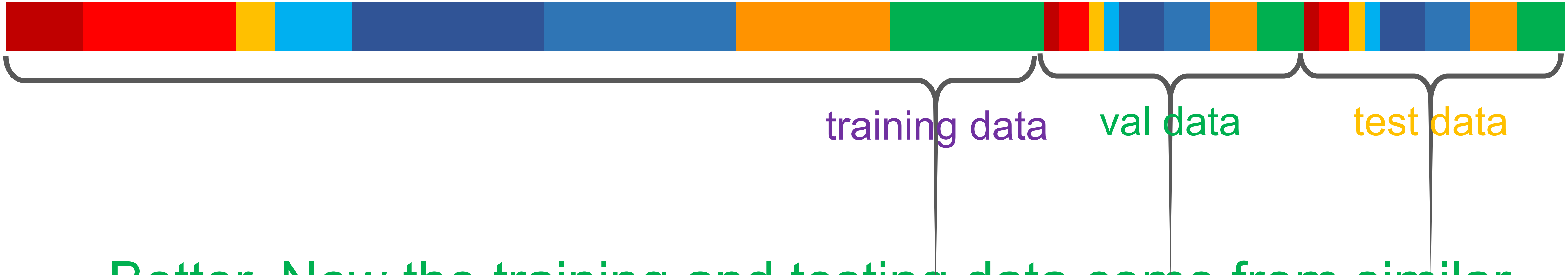
- **Idea 1:** Split the data like this:

- Is this a good way to split the data?
- **No!** Model is likely trained and tested on different data distributions

Train-Val-Test Split

- We have climate data for different countries

- **Idea 2:** Shuffle first before splitting:

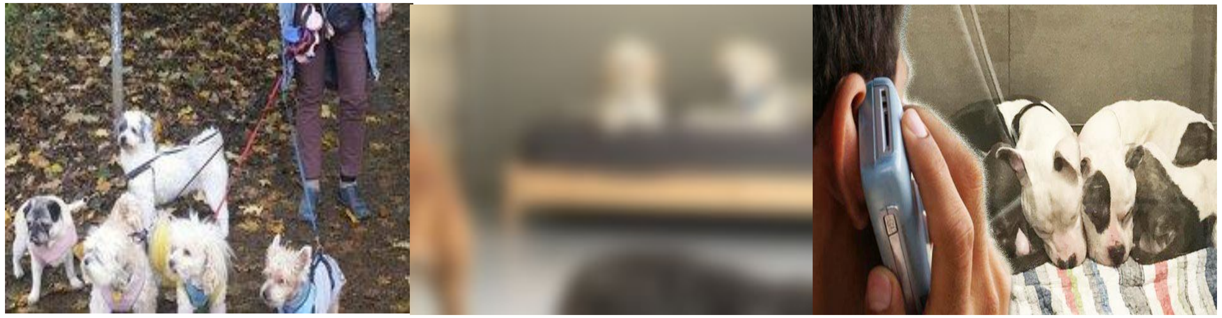


- Better. Now the training and testing data come from similar distributions.

Train-Val-Test Split

- You want to train a model for recognizing **images of dogs uploaded by users in your app**. Because you don't have enough training data from your app, you decide to **augment** it by **searching for additional images of dogs online**.

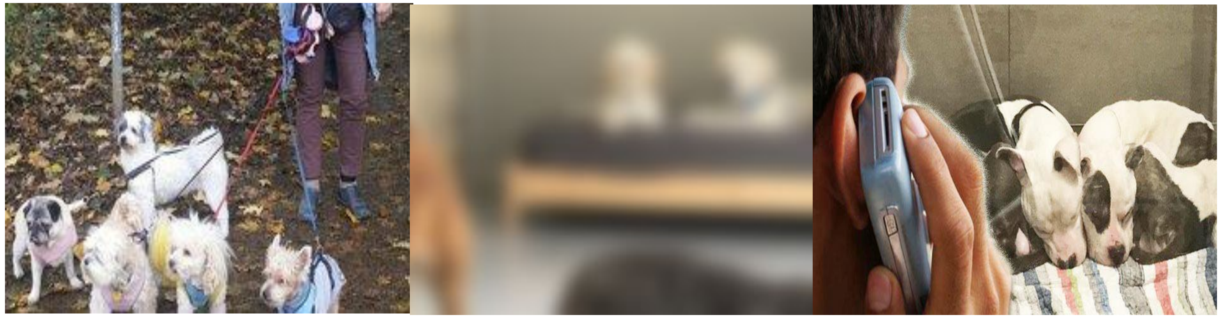
Data from the web ~200,000



Data from users / your app ~10,000

Train-Val-Test Split

Data from the web ~200,000



Data from users / your app ~10,000

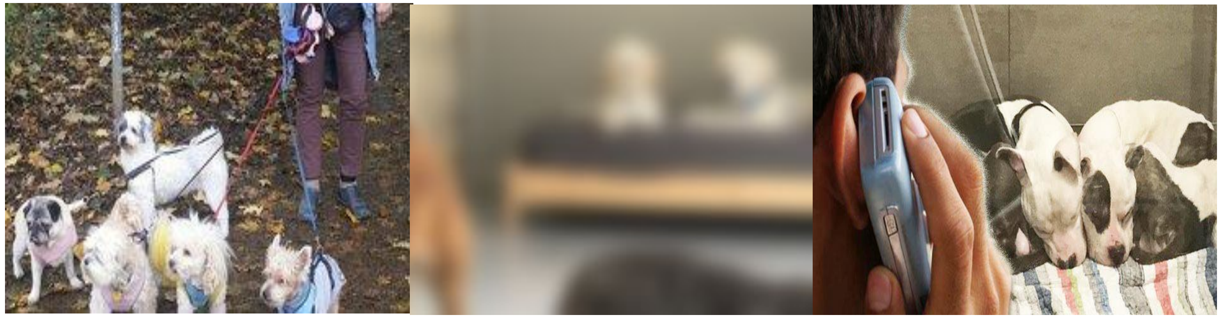
- **Idea 1:** Shuffle all the data and split

205,000 train	2,500 validation	2,500 test

- Is this a good way to split the data?

Train-Val-Test Split

Data from the web ~200,000



Data from users / your app ~10,000

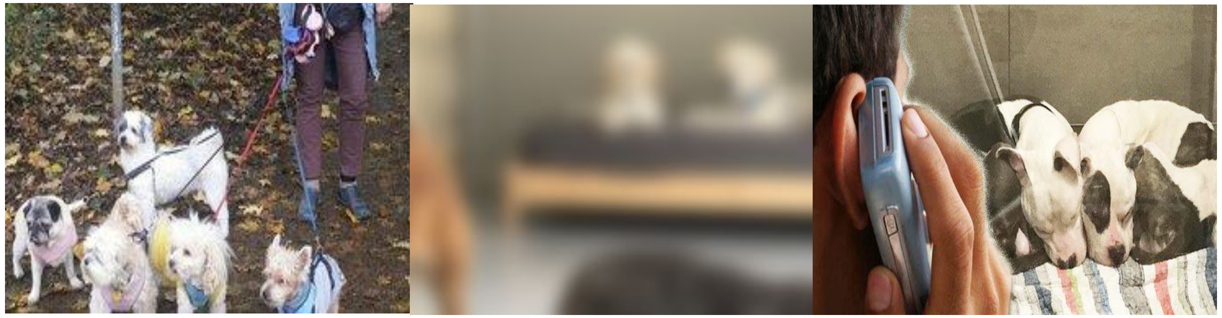
- **Idea 1:** Shuffle all the data and split

205,000 train	2,500 validation	2,500 test

- Is this a good way to split the data?
- **No!** Our model will be evaluated mainly on the overwhelming web data

Train-Val-Test Split

Data from the web ~200,000



Data from users / your app ~10,000

- **Idea 2:** Web data should only be on the training set

200,000 web data + 5,000 your data	2,500 your data	2,500 your data

- Better. Now we are targeting the data our model will be encountering in the real world.

Errors Only Happen in Deployment

- Scenario: you trained a model, and it was performing well in the test set. However, when you deployed the model as a system, it suddenly performs very poorly.

Errors Only Happen in Deployment

- Scenario: you trained a model, and it was performing well in the test set. However, when you deployed the model as a system, it suddenly performs very poorly.

- **Do a manual error analysis** between your collected data vs. the data it encountered in the real-world during deployment. Are there discrepancies?

Errors Only Happen in Deployment

- Scenario: you trained a model, and it was performing well in the test set. However, when you deployed the model as a system, it suddenly performs very poorly.

- **Do a manual error analysis** between your collected data vs. the data it encountered in the real-world during deployment. Are there discrepancies?

Errors Only Happen in Deployment

- Match your data to the real world by synthesizing

Original audio

+

Car background noise

=

Synthesized audio

Case Study: Animal Classifier

- Task: image classification
- Performance: 20% error (validation set)

- Cursory look at data shows some dogs look like other animals, and have incorrect labels

- Manually examine the mistakes
- Get ~100 mislabeled examples
- Count how many are incorrect because dogs look like other animals

- If 5/100, ~1% error will be reduced
- If 50/100, ~10% error will be reduced

Case Study: Animal Classifier

- Conduct error analysis.

image	low res	blurry	dog looks like x	poor lighting	incorrectly labelled
1	✓				
2		✓			
...		✓	✓		
n			✓		✓
% of total	12%	33%	62%	8%	59%

- Decide whether resolving these problems is needed for your model purposes.

Hyperparameter Tuning

- Selecting the best set of hyperparameters for a given training procedure.

- General idea: try out different combinations of hyperparameters and choose the **best performing model on the validation set**.

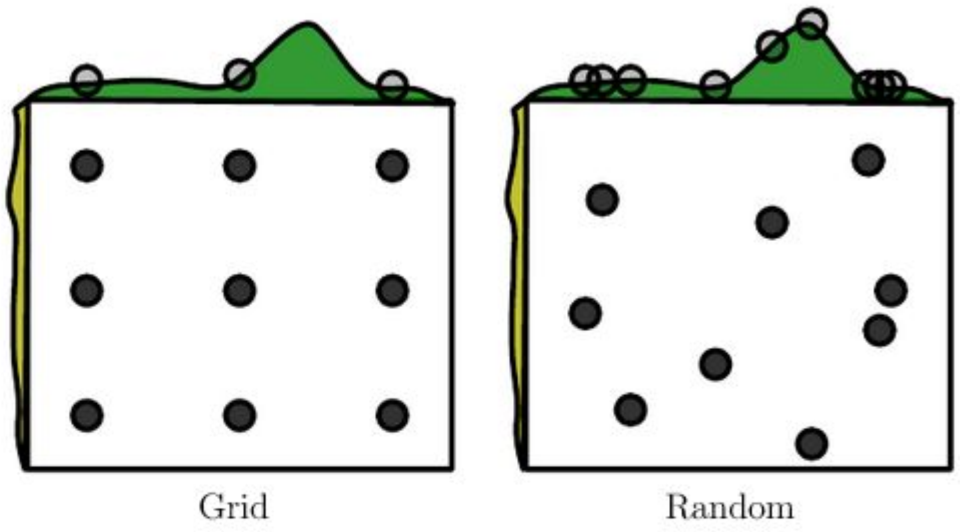
Hyperparameter Search

Size of hidden layers
linear scale (ex. uniform random from 50 – 200)

Number of Layers
linear scale (ex. uniform random from 2-6)
can be exhaustive since only few possible values

Learning Rate
log scale (ex. r = uniform random from -1 to -8, learning rate $\alpha = 10^r$)

Regularization
log scale (ex. r = uniform random from -1 to -6, regularization strength $\lambda = 10^r$)



Search from **coarse** (large ranges) to **fine** (narrow down the ranges that perform well)

General ML Pipeline

Build a simple viable initial system ASAP

Lots of noise, little structure → shallow nn
Little noise, complex structure → deep nn
    No structure → fully connected
    Spatial structure → convolutional
    Sequential structure → recurrent

Little data → Bayesian models or transfer learning

Simple structure - use what you know:
    linreg, logreg, decision trees, ada/xgboost, random forests, svm

Analyze errors and perform error analysis, decide what to prioritize next

- Determine the bottlenecks in performance
- Diagnose which components are performing worse than expected
- Determine whether poor performance is due to overfitting, underfitting, or defect in the data or software
- Repeat as needed

Suppose you are building a speech recognition system, there are lots of directions to go into: noise, accent, far/near from microphone, child’s speech, stutter