Data augmentation and scaling

UNIVERSITY OF SUSSEX

Dexter R. Shepherd, Paula Seidler, Alejandra Carriero

Recap

We have looked at many different models and classifiers

The uses of each of them for different purposes

Had a go at fitting our datasets to these models

UNIVERSITY OF SUSSEX

Contents

UNIVERSITY OF SUSSEX

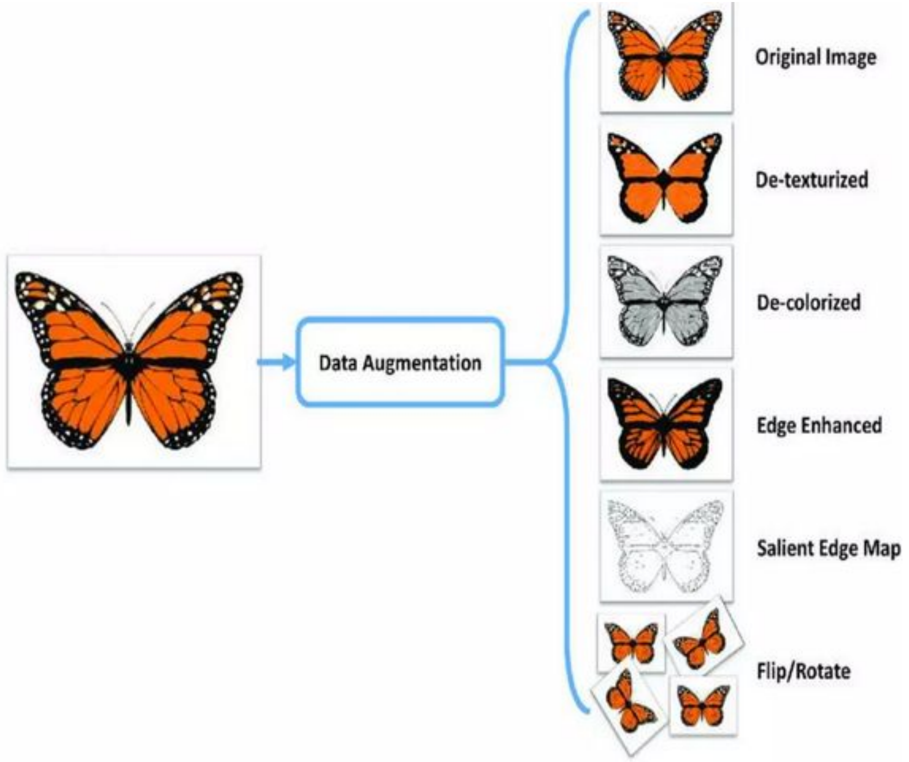
Small datasets

Having small datasets can be an issue in deep learning. These models are very complex, and having small datasets can easily lead to overfitting.

What do we do?

Data Augmentation

When we have a small dataset of images, we can **oversample** them by performing **data augmentation**.



What augmentation should I use?

This depends on your *data* and *task*, as some augmentation may negatively impact your model

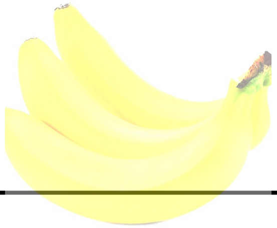
What types of data modifications preserve the original meaning for your task?

Banana detection

Detexturized

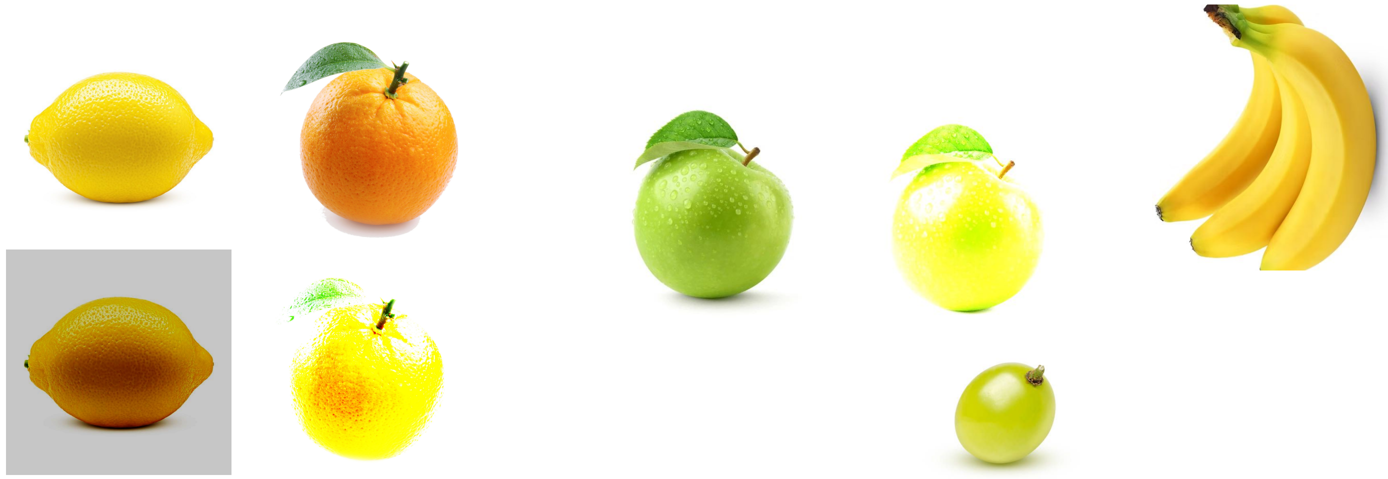
Edge enhance

Flip and rotate



A larger classifier of fruit

Maybe the lighting does matter…

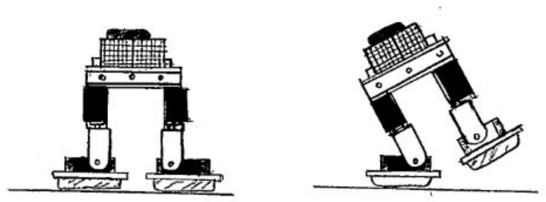


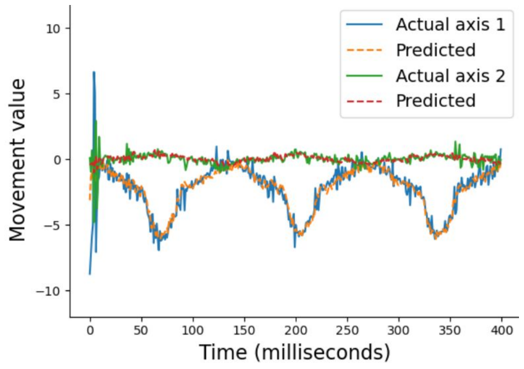
UNIVERSITY OF SUSSEX

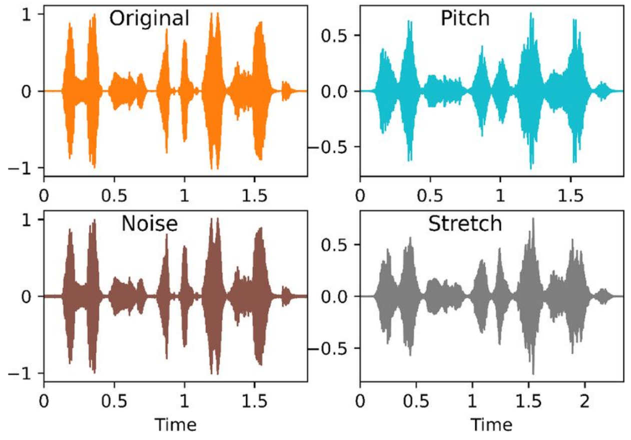
Signals

Augmentation is not limited to images…

Make sure to evaluate what data is relevant







Signals

Audio signals “hello world”

Stretch it: makes it slower spoken “heeellloooooo wwwoorrllddd”

Change the pitch augments it to different voices

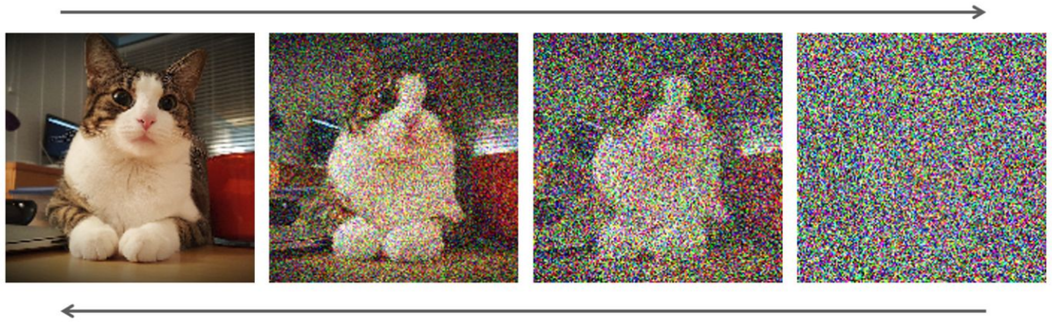
This could be helpful for voice recognition

UNIVERSITY OF SUSSEX

Adding noise

Can you rely on good image quality all the time?

Adding noise helps a model understand beyond specific colours



AI generated datasets

UNIVERSITY
OF SUSSEX

AI generated datasets

There is an approach of using AI to generate data.

Tasks where a lot of data exists such as image generation, this can be an inexpensive way of generating data

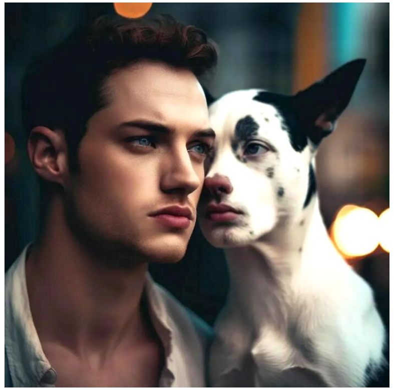
Generative AI can alter your existing images so there is a mix of truth and changes

UNIVERSITY OF SUSSEX

Problems

Bias

Data not being real enough



But they have some uses

Maybe not best for a full dataset… but can be used for transfer learning

Transfer learning

UNIVERSITY OF SUSSEX

Real world

- An artist who paints with watercolors may quickly adapt to acrylics because they understand concepts like color blending and brush strokes.
- A pianist may find it easier to learn the guitar because they already understand musical notes, rhythms, and scales.
- If you know how to bake cookies, learning to make a cake might be faster because you already understand measuring, mixing, and baking techniques.

US
UNIVERSITY OF SUSSEX

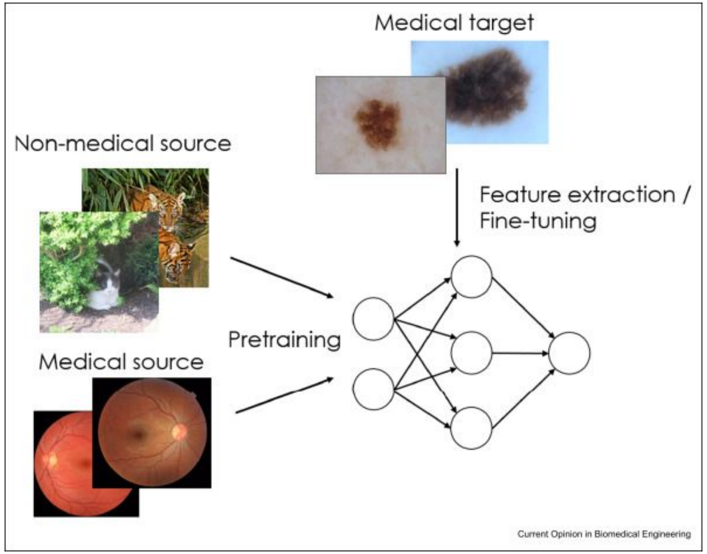
Transfer learning

If we have a small dataset, we can leverage the knowledge gained from larger datasets and have good performance on our specific task.

Transfer learning consists in freezing some layers of a pre-trained model on a previous task and training only some layers on the new task!

Transfer learning

Very surprisingly, the tasks don’t even need to be too much related!


Medical target
Non-medical source
Feature extraction / Fine-tuning
Pretraining
Medical source
Current Opinion in Biomedical Engineering

Mixing with AI datasets

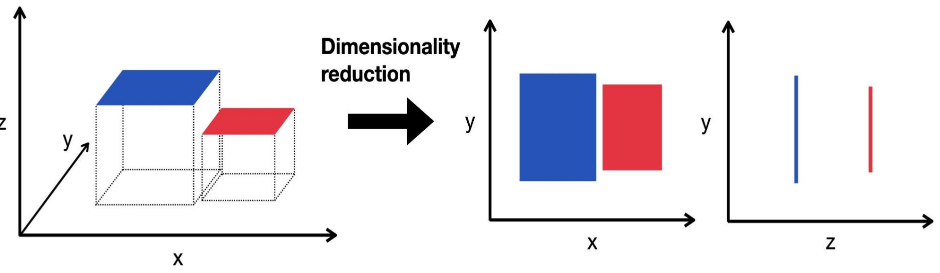
Use the AI datasets as the initial dataset

-> this helps the model understand the sort of shapes/ colours/ positions that might be relevant

Fine tune on your real dataset

Scaling

Dimensionality reduction



What if we can't plot our data, or when we do there is no clear separation?

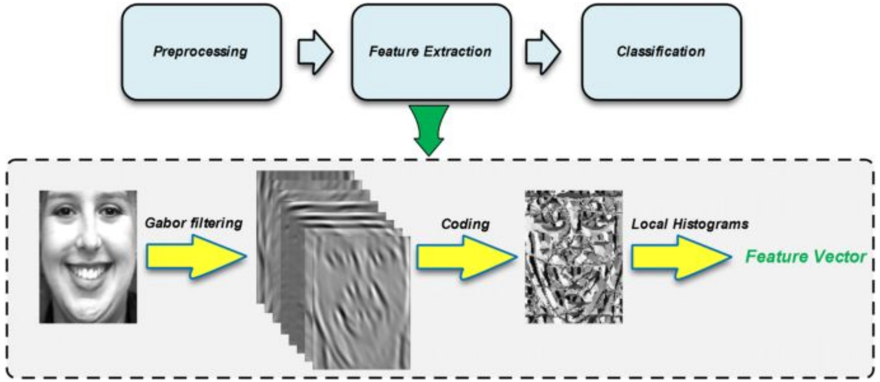
Examples:

- Images
- High dimensional such as multiple sensors (which sensors are useful?)
- Biological data
- Reducing too much data can come at the expense of accuracy

Example

A Gabor filter is a linear filter used in image processing for edge detection, texture classification, feature extraction and disparity estimation

By filtering the image we have remove a lot of unnecessarily information for our model



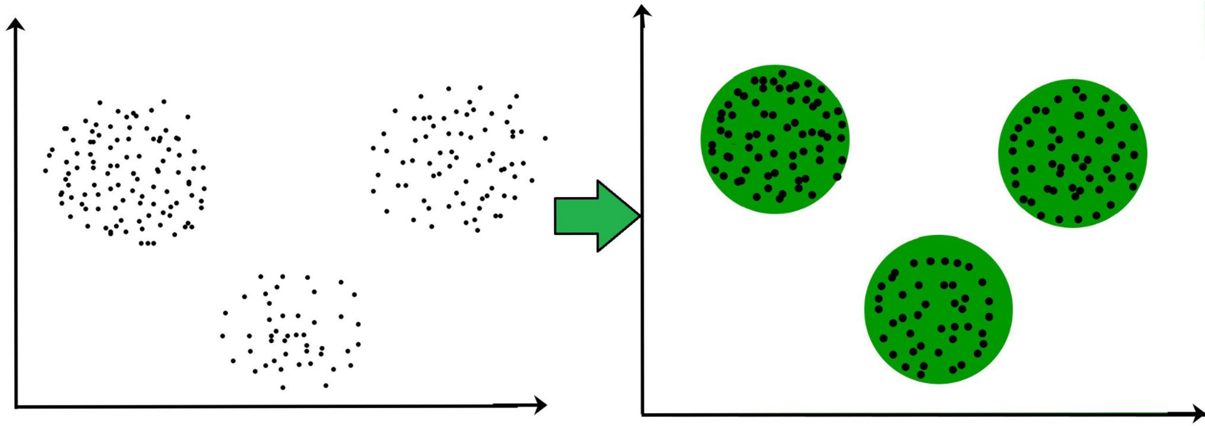
Clustering

Dividing the unlabeled data or data points into different clusters

Types of clustering

- Association models
- Centroid models

There are others but we are not covering this

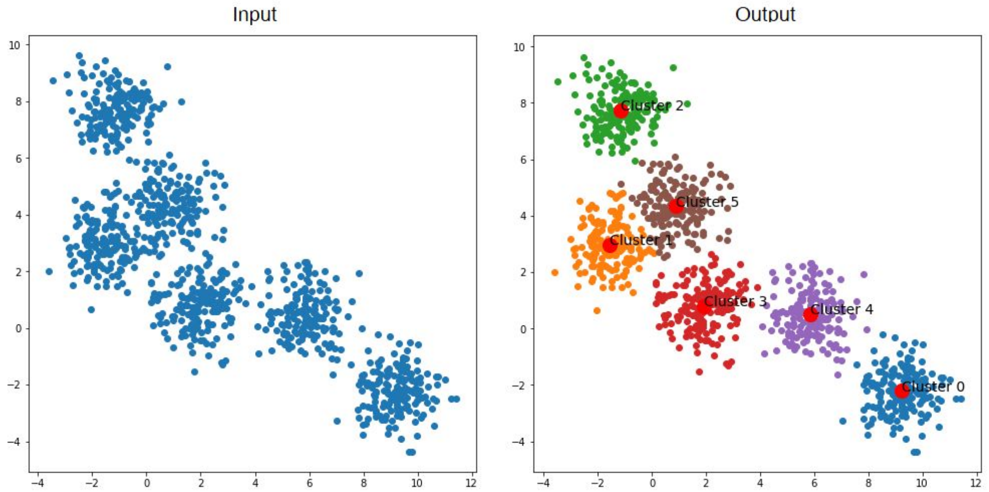


Centroids

Refers to the central point of data

These centroid points are calculated and used to determine where a point of data lies within the clustering

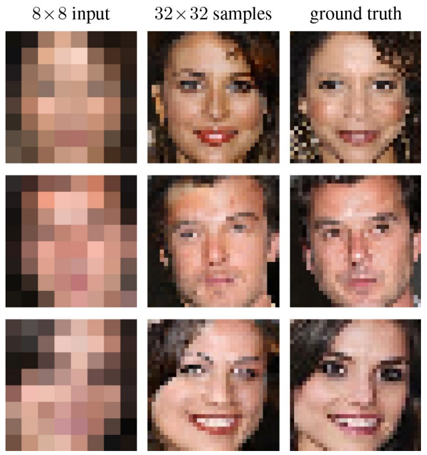
Clustering aka K-nearest neighbours

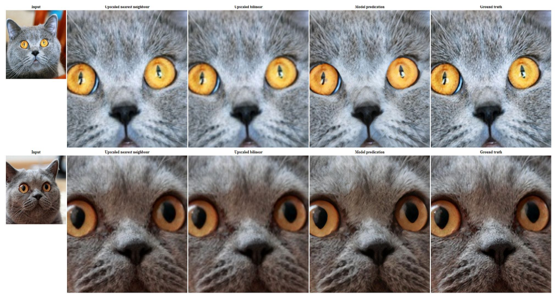


Compression

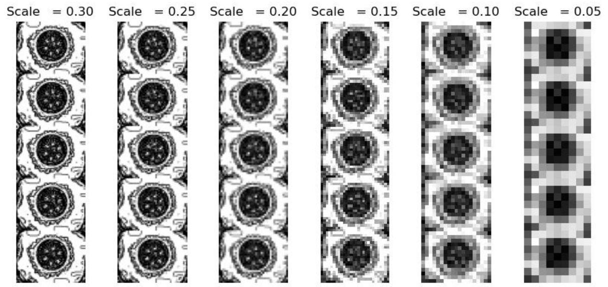
How compressed can your data set be and still provide useful information?

Depends on the task





Segmented Image when K = 6

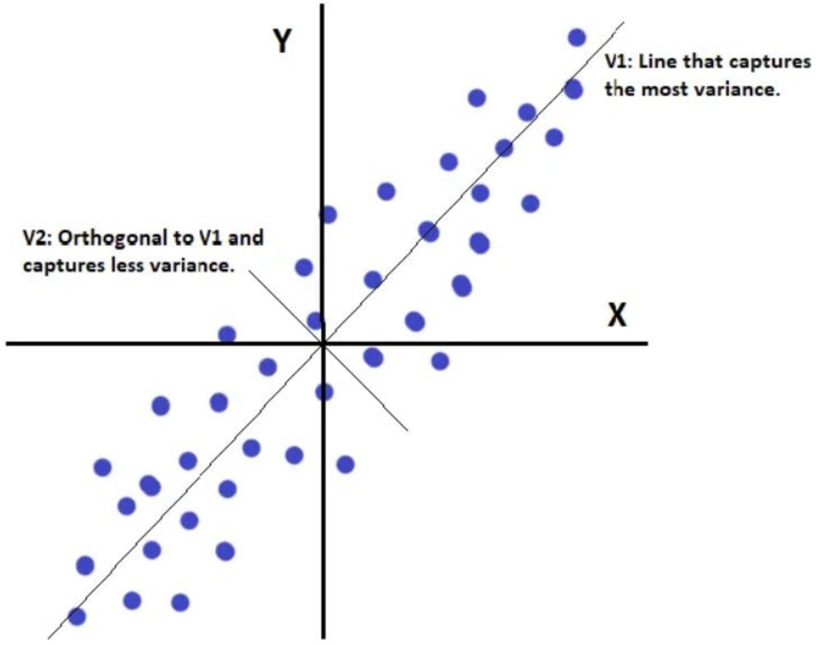


Principal component analysis

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables.

In simple terms it is reducing the number of variables from a data set - potentially at the cost of accuracy

This method helps in identifying the directions (or components) in which the data varies the most

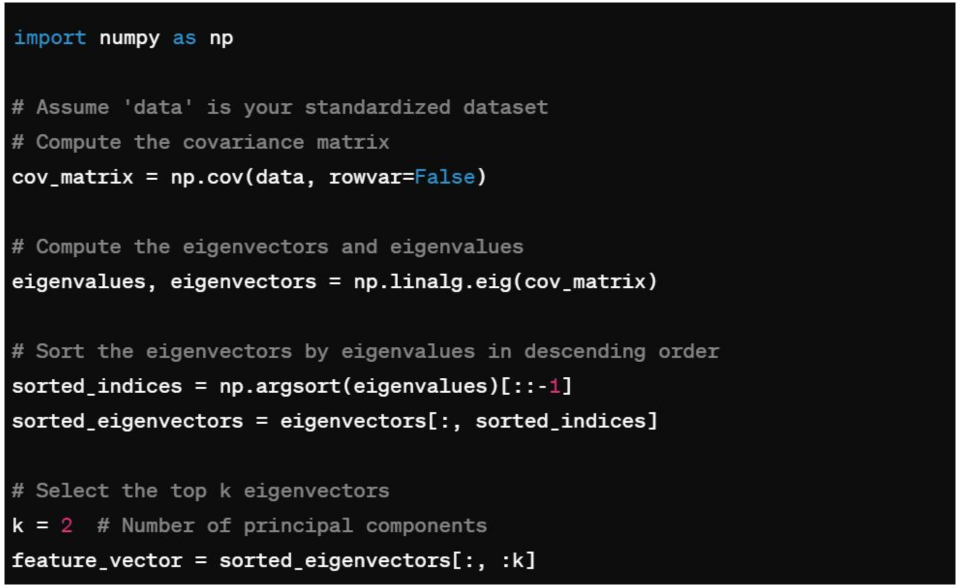

Y
X
V1: Line that captures the most variance.
V2: Orthogonal to V1 and captures less variance.

Principal component analysis

Steps:

1. **Standardise the range of continuous initial values:**
   - This means centering the data by subtracting the mean and scaling it by dividing by the standard deviation. Standardization ensures that all features have equal importance in the analysis.
2. **Compute the covariance matrix**
3. **Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components:**
   - In PCA, eigenvectors represent the directions along which the data varies the most.
   - An eigenvalue (λ) represents a scalar that indicates how much variance is explained by the corresponding eigenvector. In PCA, eigenvalues quantify the importance of each principal component.
4. **Create a feature vector to decide which principal components to keep**
5. **Recast the data along the principal components axes**

Coded solution



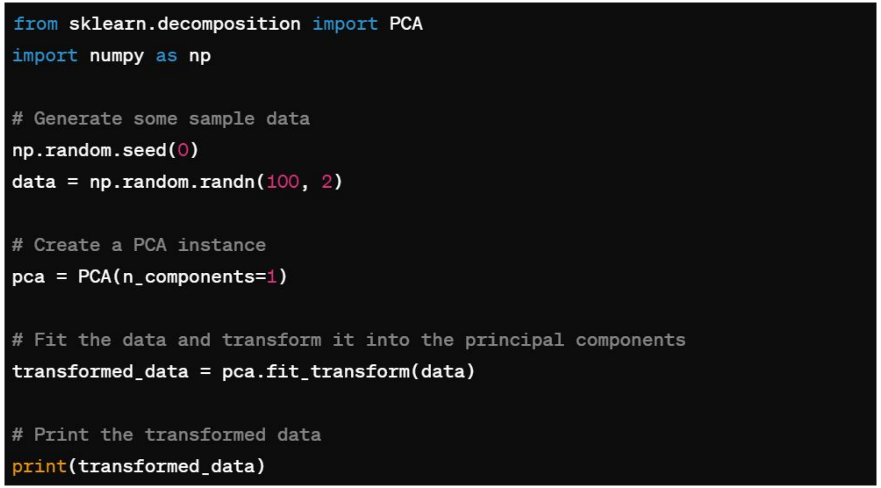
```python
import numpy as np

# Assume 'data' is your standardized dataset
# Compute the covariance matrix
cov_matrix = np.cov(data, rowvar=False)

# Compute the eigenvectors and eigenvalues
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# Sort the eigenvectors by eigenvalues in descending order
sorted_indices = np.argsort(eigenvalues)[::-1]
sorted_eigenvectors = eigenvectors[:, sorted_indices]

# Select the top k eigenvectors
k = 2  # Number of principal components
feature_vector = sorted_eigenvectors[:, :k]
```



```python
from sklearn.decomposition import PCA
import numpy as np

# Generate some sample data
np.random.seed(0)
data = np.random.randn(100, 2)

# Create a PCA instance
pca = PCA(n_components=1)

# Fit the data and transform it into the principal components
transformed_data = pca.fit_transform(data)

# Print the transformed data
print(transformed_data)
```

Choosing a method

Choosing approaches to data depend on your data

Is your input to the model a 2D vector? Reducing the dimensionality is not useful.

Have you plotted the data, are there obvious trends within classes?

Is your input data the same size each time?

Data set 1

You have a table of data with sensor readings from a number of sensors over the year. We want to predict weather based on these sensors.

It seems not all sensors are relevant.. How can we extract only relevant information

UNIVERSITY OF SUSSEX

Data set 2

We have MRI scans of torsos and want to detect if there is cancer in the heart.

Can we reduce the size of this more?

Data set 3

We have images of grass over the year

We want to see when it looks the driest

Can we reduce the data to have a smaller model, if so what can we use?

UNIVERSITY OF SUSSEX

Conclusions

Augmenting data can help improve model effectiveness when training models with small datasets - but does not work for every scenario

Transfer learning can overcome some of the issues caused by augmentation

Reducing the dimensionality can lead to better and smaller models

When picking a model, or dimensionality reduction technique we can pick this based off inspecting the data and experimentation

Much of this comes with time and experience

UNIVERSITY OF SUSSEX

Labs

UNIVERSITY OF SUSSEX

Using existing datasets

Using your existing datasets, establish methods of data augmentation that work for your data

Hint: google scholar will likely have existing literature

Questions

How does augmenting your dataset improve the model accuracy (both train and test)?

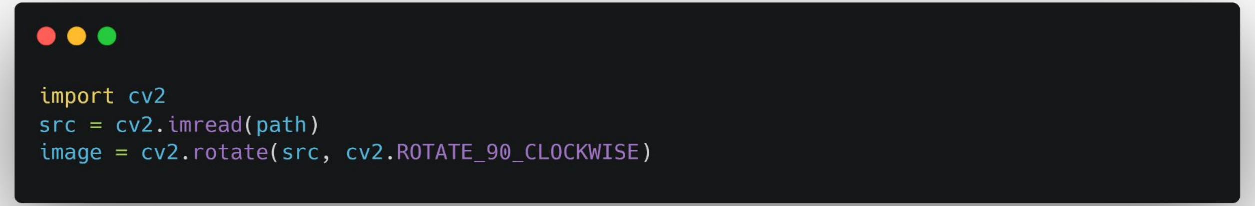
Is there an ideal ratio between augmented data and real data?

UNIVERSITY OF SUSSEX

Useful functions for images

Image filters:
https://learnopencv.com/image-filtering-using-convolution-in-opencv/

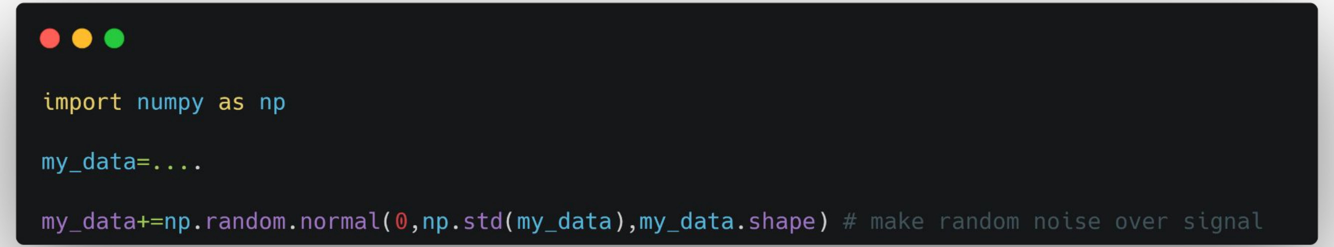
Rotations:



```
import cv2
src = cv2.imread(path)
image = cv2.rotate(src, cv2.ROTATE_90_CLOCKWISE)
```

Useful functions for signals

Adding gaussian noise:



```
import numpy as np

my_data=....

my_data+=np.random.normal(0,np.std(my_data),my_data.shape) # make random noise over signal
```