Best Practices for building Deep Learning based Image classification systems

STTP on Research Issues and Challenges in Deep Learning based Medical Image Analysis and Diagnosis
Hosted by Sri Ramakrishna Engineering College,
Coimbatore – 641022
TamilNadu
India

Presenter:
Rajesh Thennan
https://thennan.me

Presentation content and code available at:
https://github.com/rthennan/sttp-srec-aug2020

Agenda

- How to get started with Deep Learning?
- High level classification of Machine Learning models
- Deep Learning , TensorFlow, Python – Why?
- Workflow for building an Image Classification System
- Challenges in Medical Image Data acquisition
- Choosing the best type of data
- Exogenous variables / demographic info
- Improve Preprocessing performance
- Data Preparation best practices
- Convolutional Neural Networks (CNN)
- Concatenated Model
- Building and Training a CNN based Image classifier
- TensorFlow Callbacks
- Overfitting
- Model Deployment
- Possible future

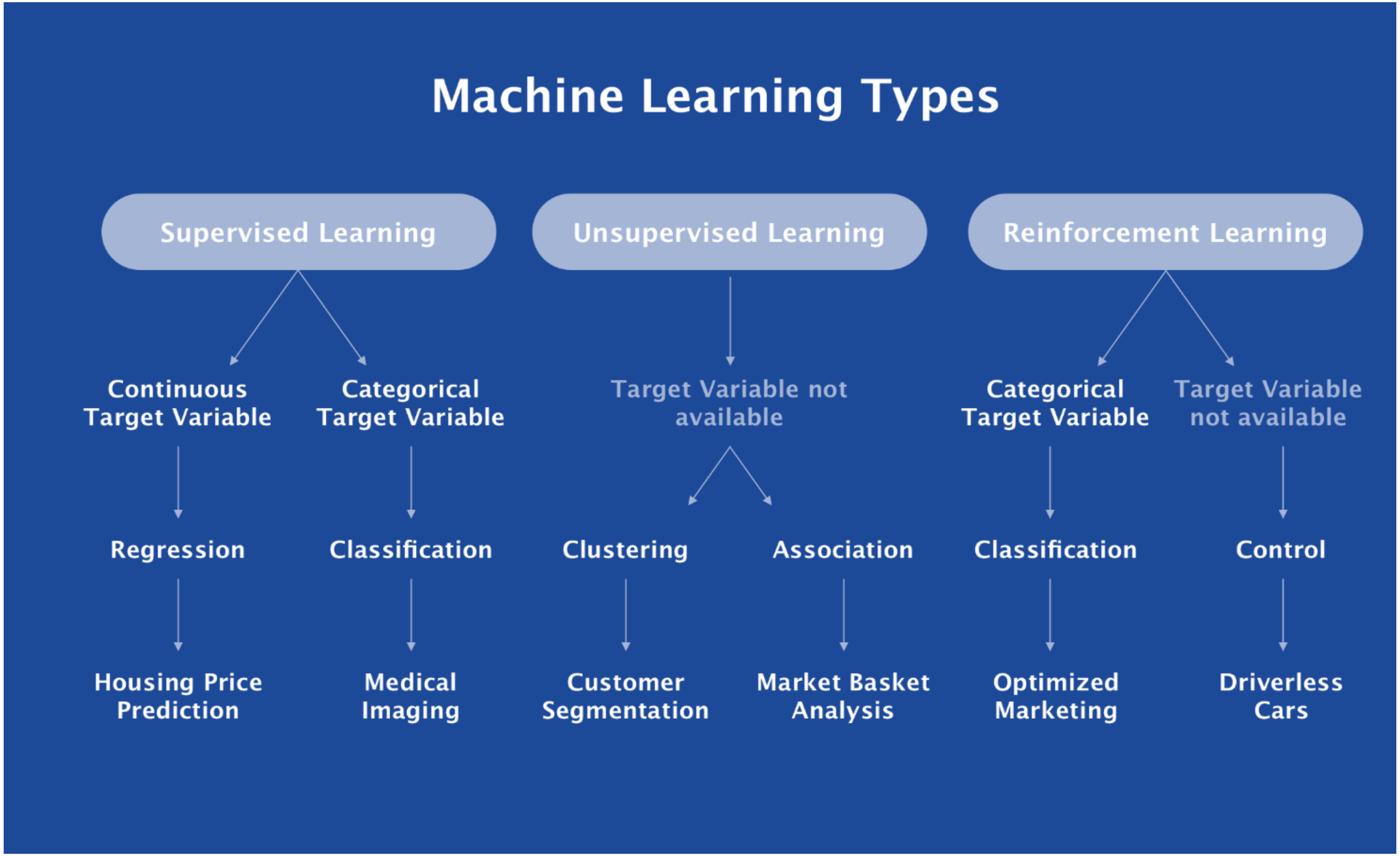
How to get started with Deep Learning?

- **Google Colab**
  https://colab.research.google.com/
  - Free runtime environment for Python with GPU and TPU
  - Allows adding session storage (Temporary) and mounting Google Drive
  - Can be shared with collaborators
  - TensorFlow in google colaboratory (YouTube Playlist)

- **Public Datasets**
  - Kaggle:
    - https://www.kaggle.com/kmader/siim-medical-images
    - https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
    - https://www.kaggle.com/paultimothymooney/kermany2018
    - https://www.kaggle.com/tawsifurrahman/covid19-radiography-database
    - https://www.kaggle.com/nih-chest-xrays/data
    - https://www.kaggle.com/praveengovi/coronahack-chest-xraydataset
    - https://www.kaggle.com/sulianova/cardiovascular-disease-dataset
    - And much more…
  - Consolidated list from Dr. Andrew L. Beam:
    - Medical Data for Machine Learning
  - ODSC - Open Data Science's Medium article:
    - 15 Open Datasets for Healthcare

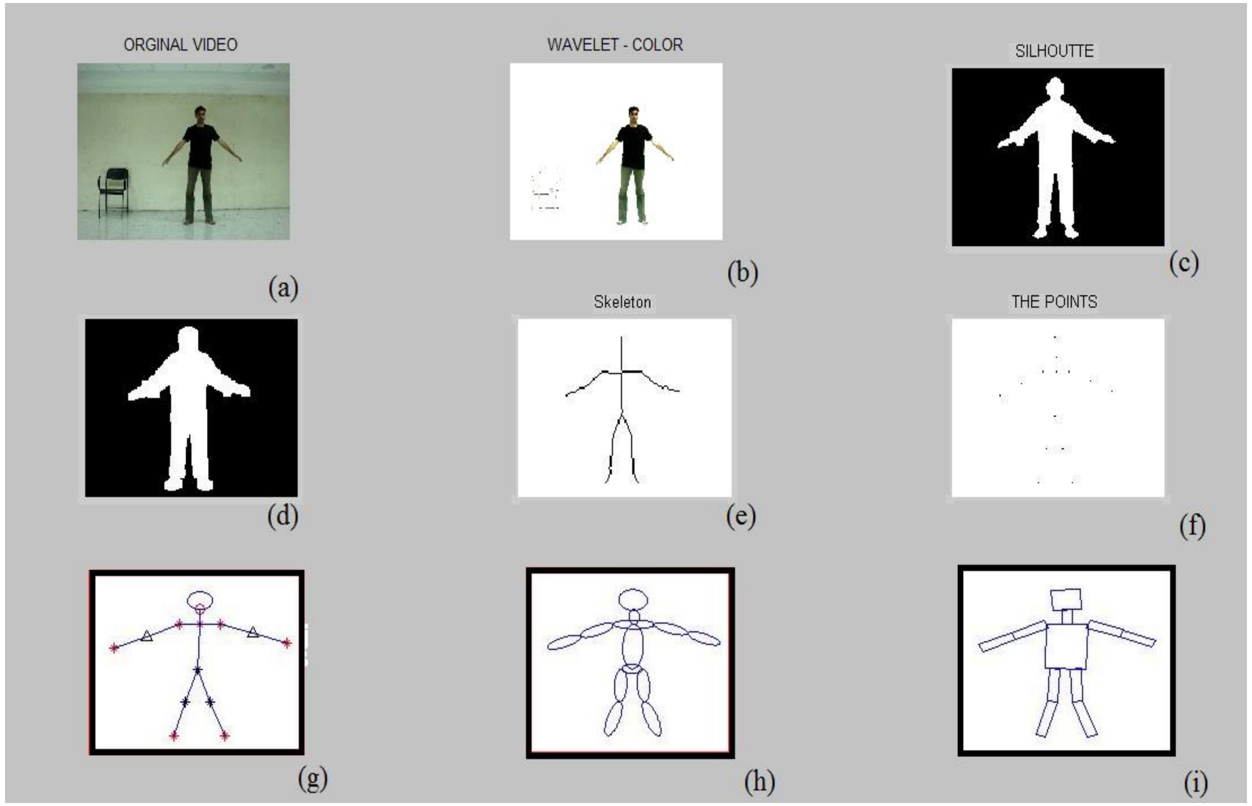
- **Learning Resources:**
  - YouTube:
    - Python 3 basics (Playlist)
    - Python Full Course - Learn Python in 12 Hours
    - Deep Learning with TensorFlow 2.0
  - Udemy:
    - Deep Learning A-Z™: Hands-On Artificial Neural Networks
    - TensorFlow 2.0: Deep Learning and Artificial Intelligence
  - Coursera:
    - Python for Everybody Specialization
    - Deep Learning Specialization (Andrew Ng)

High level classification of Machine Learning models



Deep Learning , TensorFlow, Python –Why?

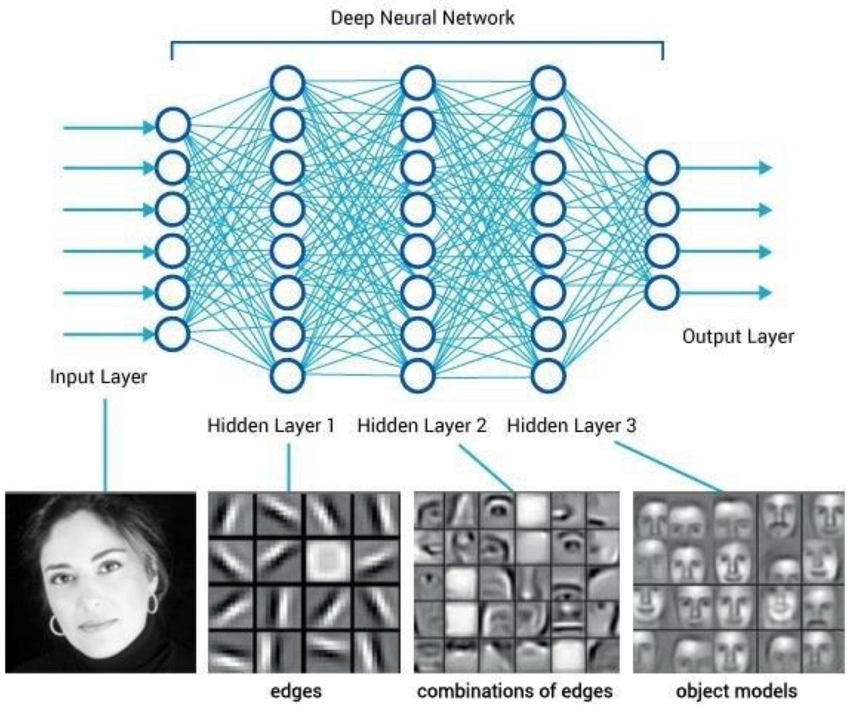
- Why Machine Learning?



(a)- original video, (b)- output of feature extraction, (c)- Silhouette,
(d)- Dilated Silhouette, (e)- output of thinning,
(f)- accumulation of the points extracted, (g) to(i)- 2-D models

Deep Learning , TensorFlow, Python –Why? (Contd.)

- Why Deep Learning?
  - In traditional Machine learning techniques, most of the applied features need to be identified by a domain expert in order to reduce the complexity of the data and make patterns more visible to learning algorithms to work.
  - Deep Learning algorithms try to learn high-level features from data in an incremental manner. This eliminates the need of domain expertise and feature extraction.
  - And in today's Big Data Era, data is abundant and that is exactly what Deep Learning models are good at.



Deep Learning , TensorFlow, Python –Why? (Contd.)

- Why TensorFlow?

- Why Python?
  - Free
  - Simple and Easy to learn
  - Extensive selection of libraries for Machine Learning
  - OS independent
  - Google Colab

Workflow for building an Image Classification System

Data Acquisition
- Onetime
  or
- Ongoing

Cleanup

Preprocessing
- Denoise
- Enhance
- Resize (dimensions)
- Scale (pixel level)

Train – Test split

Preparation
- Shuffle
- Balance
- Sequence
- Reusable prepared file

Build and Train Model
- Base Model
- Hyper parameter tuning
- Choose Optimal model

Deployment
- Standalone
  or
- Server

Challenges in Medical Image Data acquisition

- **Data Privacy and Security**
  - There are **ethical and legal obligations** for health care providers **to preserve the privacy and confidentiality** of patient information, which can contain some of the most intimate information conceivable about an individual.
  - **Mitigation:**
    Develop a data masking / scrambling system that data providers can use to reliably mask Personally Identifiable Information (PII)

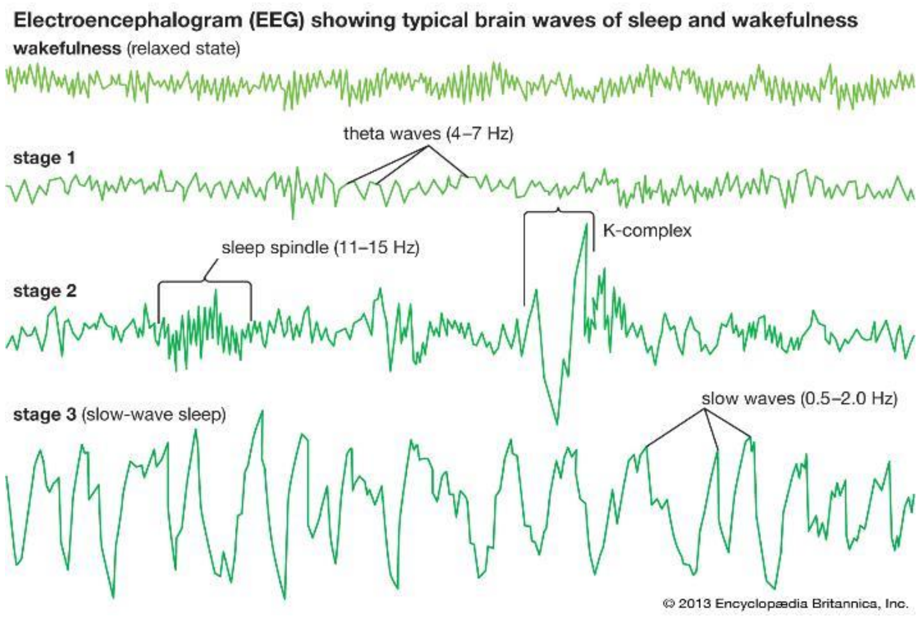
- **Tagging / Classifying data at source**
  - All data is **not classified at source**
  - We might have to work with the Medical personnel like Doctors and Radiologists at the data source to classify the data before receiving it from them.
  - Considering the Data Privacy concern, they might not allow a Researcher / Data Scientist even to look at unmasked data.
    Mitigation:
    Develop an interface / system that can be used by such personnel directly to classify their data , before or after masking PII.

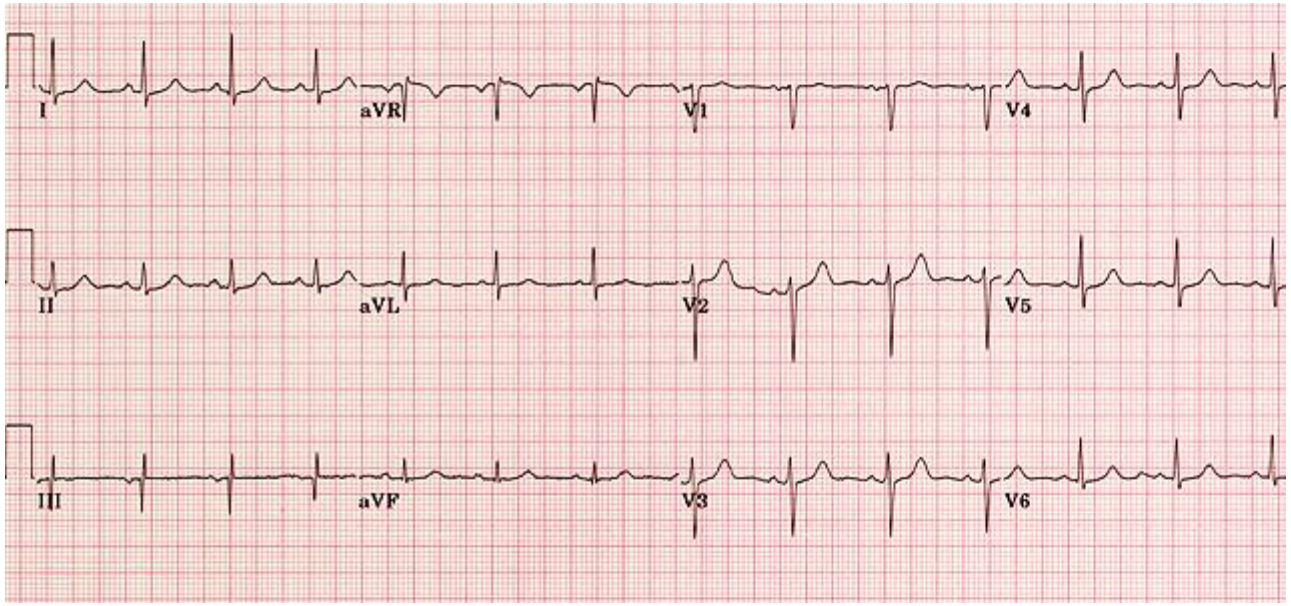
- Most hospitals and Medical Labs have dedicated IT staff.
- But they are minimally staffed and focus mainly on maintaining the IT infrastructure and regular maintenance activities.
- Hence, we cannot expect their assistance to perform the above activities.
- So the tools we provide must be as user-friendly and intuitive as possible.

Choosing the best type of data

- Print an Audio Signal and try to process it as an image?
- Print a grayscale image as an intensity distribution (0-255) for human interpretation?
- Print Stock prices as graphs / candles and process them as Images?

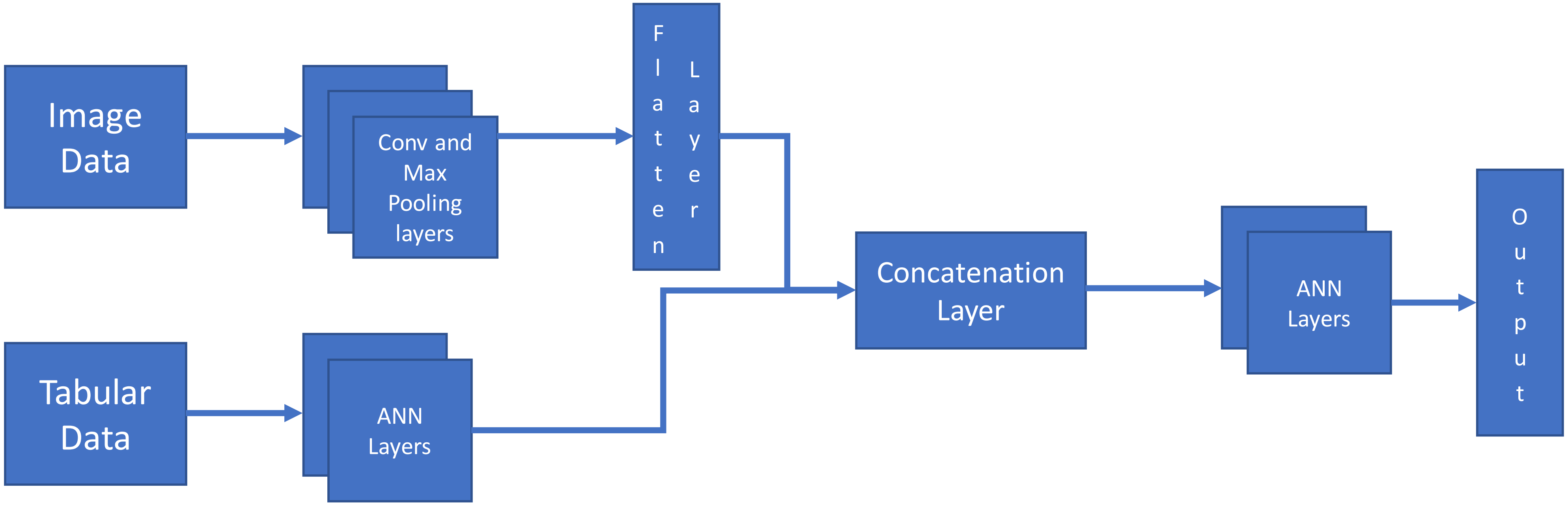
Why process the below as images then?:





Exogenous variables / demographic info

- In Image Analysis, data scientists / researchers working in Deep Learning **usually tend to ignore demographic information, even if available.**
- This is primarily because they either think the information is useless compared to the Images or because they are **unaware of the techniques to include them in the Image Processing model.**
- Generally, there are **no explicit PCA / feature engineering or comorbidity** checks done in deep learning. So, they wouldn't have even had the opportunity of checking the relevance.
- **Demographic info is critical** and should be included in the model's design (Concatenated Models).



Improve Preprocessing performance

- A powerful GPU could have a multifold performance improvement while training deep learning models.
- But a **Powerful, multi-core CPU** could **help speed up the rest of the process**, provided:
    1. The Programing language supports parallel processing
    2. Code is optimized / rewritten for the same

- We are unlikely to get 100% scaling for each additional CPU thread:
    - Clock speed of a CPU while running a single thread process would be higher, compared to a higher CPU utilization % with parallel processing.
    - This is because most modern CPUs have a different base and different boost clock.
    - We might face thermal throttling with higher CPU utilization and in turn lower clock speeds, also impacting the CPU IPC (Instructions Per Cycle).
    - There are additional overheads due to thread initialization, resource allocation and data read.

Challenges:

- Increased code complexity.
- Might need additional file handling and data processing due to Interprocess Communication limitations.
- In simple / basic operations, the time taken for the additional overheads, might be higher that the time taken to complete the task sequentially.

	Parallel processing / multi-threaded (30 threads)	Sequential / single- threaded
Project1 (Image)	00:05:22 322 seconds (~12 times faster)	01:07:00 4020 seconds
Project2 (Time Series)	~ 8 hours	~ 80 hours (Estimated)
Chest X-Ray Dataset	41 seconds	73 seconds

Examples of performance improvement through CPU parallel Processing for data preprocessing

Data Preparation best practices (Classification tasks)

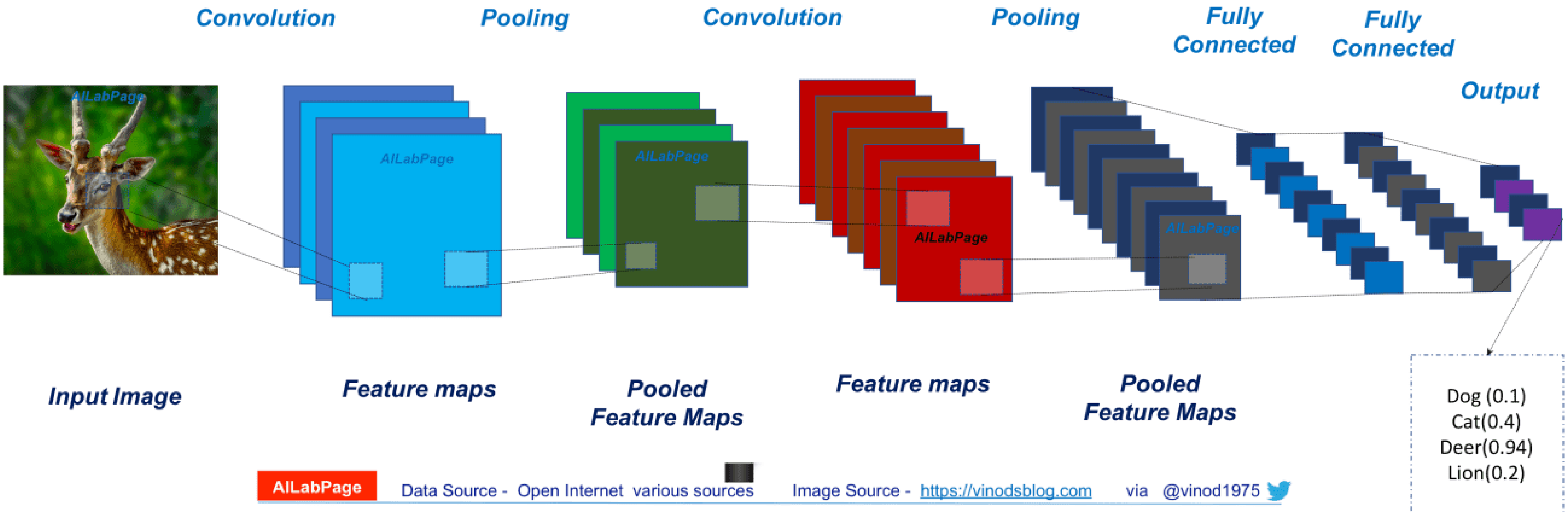
- Shuffle:
  - Always shuffle the data to avoid any accidental trend introduced either during Data acquisition or during pre-processing
  - This also helps the Deep Learning model's learning rate

- Balance Data
  - **Imbalanced data in multi class classification** can cause the model to **train in an undesired manner**.
  - In Medical data, positive candidates would usually be much lesser than the negative candidates.
  - For example:
    If we train a model with a cancer dataset with 97 % benign data and 3% tumour data, the model just has to predict everything as benign to get a 97% accuracy.

- Sequential / Alternating arrangement after balancing (optional):
  - Rebuilding the original dataset using the balanced data, by alternatively arranging them.

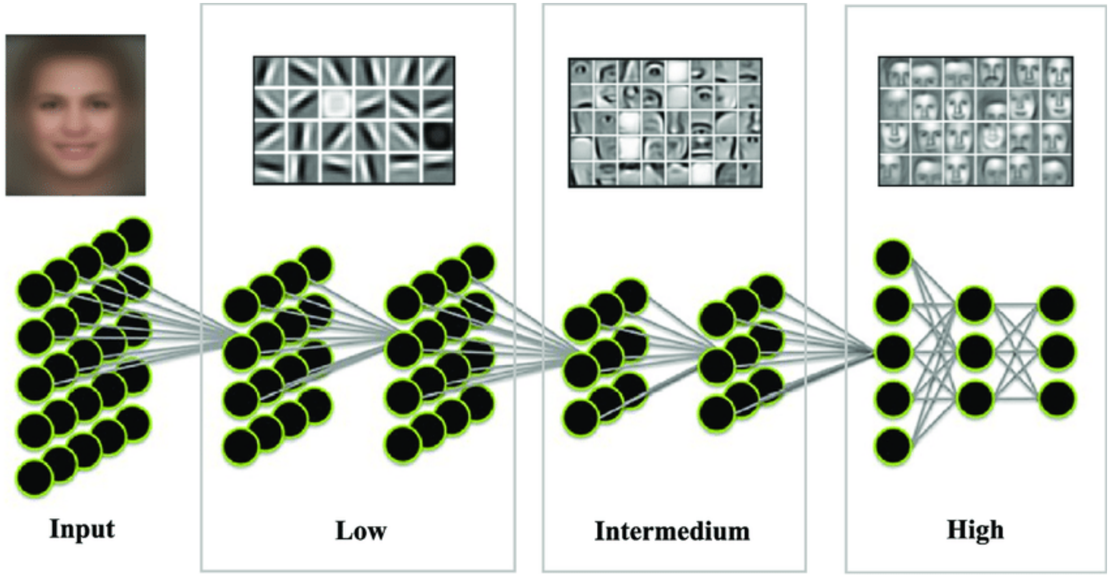
Convolutional Neural Networks (CNN)

A Class of deep neural networks, most applied to analyzing visual imagery



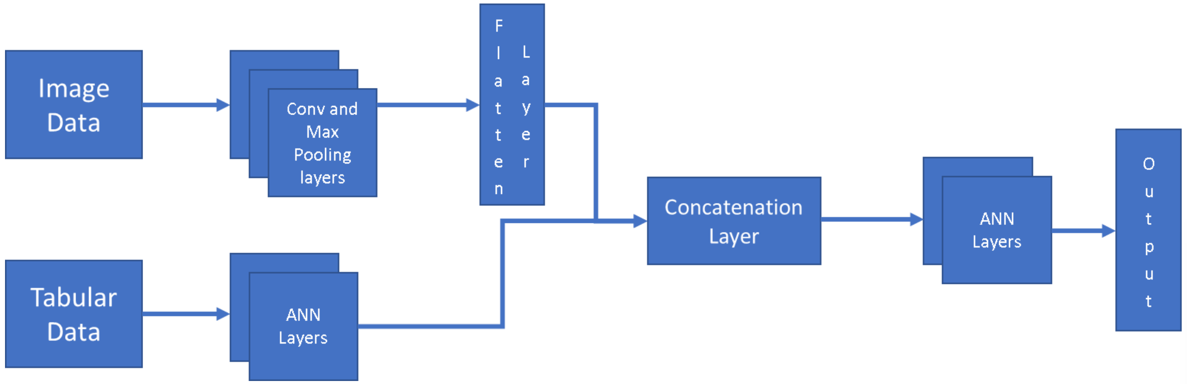
Convolutional Neural Networks (Contd.)

Feature extraction in CNN



[2D Visualization of a Convolutional Neural Network for MNIST]()

Concatenated Model




```
Layer (type)                    Output Shape          Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 127, 127, 1) 0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 127, 127, 64) 640         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 127, 127, 64) 256         conv2d_1[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 2)]          0
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 63, 63, 64)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 10)           30          input_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 63, 63, 64)   0           max_pooling2d_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 10)           0           dense[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 254016)       0           dropout_1[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 254026)       0           dropout_2[0][0]
                                                                 flatten[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 10)           2540270     concatenate[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 10)           0           dense_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 10)           40          dropout_3[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 2)            22          batch_normalization_2[0][0]
==================================================================================================
Total params: 2,541,258
Trainable params: 2,541,110
Non-trainable params: 148
```

TensorFlow Callbacks

- tf.keras.callbacks

- Callbacks I mostly use:
  - TensorBoard:
    - Allows visualizing the model’s accuracy and loss while it is still training.
    - Can also be used to compare the efficiency of different models in the same page
      tensorboard --logdir=‘<logDirectoryName>’

  - ModelCheckpoint:
    - Allows saving models from each epoch and not just the final one.
    - These models can be differentiated by the epoch number, validation accuracy and loss

Model Overfitting

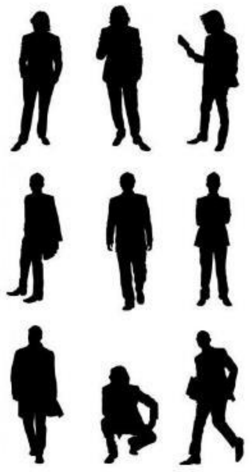
- Undesired.
- Check the Validation accuracy and not just the training accuracy.
- A model with high training accuracy but low validation accuracy will perform poorly during inference.
- Happens when there are too many trainable parameters or too little data to train on.
- Too many epochs while training could also eventually result in overfitting.
- Check the number of trainable weights using model.summary() and reduce them by simplifying the model.
- Adding dropout and regularization could help to an extent.

Model Deployment

- Serving Models using Tensor Serve
  - Always running
  - Doesn’t have to be initialized and loaded for every prediction
  - More efficient in answering multiple, parallel requests

- Challenges:
  - Accepts only json data
  - Input images / data have to be scaled before being sent as a request
  - Response will be probability distribution
  - Mitigation:
    Create simple middleware using that does the following:
    (ex: REST API server with Flask)
    - User friendly interface to Browse and select files
    - Perform all pre-processing steps used while training
    - Convert to numpy stack and reshape
    - Convert to json.dumps
    - POST request the tensor serve
    - Convert the response received into labelled class (argmax + dictionary lookup) and then send this response to the user

Possible Future



Bio Signals

- Discontinuous and sparse data
- Scope of improvement for sensor adhesion and efficiency

Gesture estimation

Medical Image

Audio / Speech

- High potential , low research

Concatenate

Alert Medical Personnel or Prescribe Medication

Questions?

Thank You

Presentation content available at:
https://github.com/rthennan/sttp-srec-aug2020

Rajesh Thennan
https://thennan.me
E-mail:
rajesh.Thennan@gmail.com