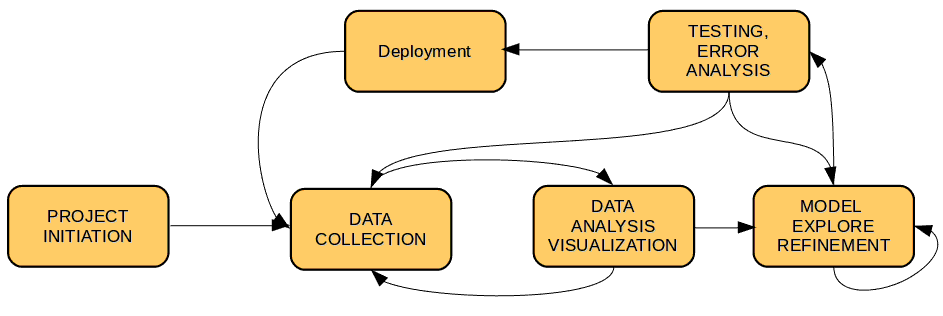
**Executing ML/DL Projects**

The goal of this document is to provide a framework for executing ML/DL projects that can be referenced by practitioners.

**ML/ DL Project Life Cycle:** ML/DL projects are highly iterative, below is the life cycle will discuss each stage in detail.



A ML/DL Project can be broadly broken into these sub-tasks.

1. Building Data Pipeline to Load, Clean and Process Data
2. Data Refinement and Visualization
3. Train, Build and Deploy Model
4. Integrate ML/DL with the application. Example – Rest API, IOT device porting.

**Project Initiation:** Below are the Checkpoints

1. Do all stakeholders understand the very nature of Machine Learning (or Deep Learning) process and projects ?
2. Do all stakeholders have the same understanding of definitions of various ML performance metrics like – Accuracy, Precision, F1 Score, etc. ?
3. Is the acceptance criteria well defined (quantitatively). Below are the POINTS we should consider.
   * 1. Cost of data acquisition. ex- How expensive is Data collection and labeling in terms of both [TIME + COST].
     2. Computational resource available + Training time.
     3. Model Measurement Metric (ex- F1score, Accuracy). Any exception in standard definition example *>90% confidence will be considered as Valid Prediction.*
     4. In general if N metric, 1 optimizing and N-1 will be satisfying. Example *Optimize a detector for F1 score while satisfying the conditions inferencing time on Nvidia Jetson Nano to >15fps and max memory taken <50 mb.*
     5. Time and memory taken for inference.
4. **Regularities:**
   1. Version code and Data-set differently. [Explore [dvc](https://dvc.org/)]
   2. Make sure framework and it’s version across the team is same.

**Data Collection:** It is advisable distribution for Train and Test set should be as close as possible. Below are the few checkpoints which we should consider while Data Collection.

1. List down all variables in the data. Example – For a computer vision project data can have variables like: Lighting Condition, Occlusion, Noisy Image, Low Resolution Image, Geometrical Transformations etc. Capture all these variables as much as possible.
2. Never collect all data in ONE GO. Data Collection and Data Analysis should be an iterative process.
3. Vary Time, Sensor, Environment while collecting data.

**Data Analysis:** Collect some domain knowledge, analyze data for below anomalies.

1. Data distribution and outliers, use t-sne algorithm.
2. Scaling and Normalization
3. Representation for Categorical Variables
4. Class Imbalance
5. Missing Values

**Data Split Ratio:** Ratio of Train/ Dev(Validation)/ Test

Data-set size is less example 10K traditional split should be followed.

|  |  |  |
| --- | --- | --- |
| Train (60%) | Dev/ Validation (20%) | Test (20%) |

If you have larger data-set size say 1 Million (10,00000) and more below split is favorable.

|  |  |  |
| --- | --- | --- |
| Train (98%) | Dev/ Validation (1%) | Test (1%) |

*General rule can be 100-2X, X, X if X amount is approximately 10K else 60, 20 , 20.*

**Addressing Different Train/Test Distribution:** It is recommended that train and test data should have similar distribution. Where it might be case where we can have subtle difference between Train/ Test distribution. Example Cat vs Dog for training we have collected images from the Internet which are in High Resolution where as for testing users are uploading images from Mobile Camera which is full of Blur, Occlusion and Different Lighting condition.

Lets formulate the problem: Say you have below kind of data to start with.

|  |  |
| --- | --- |
| 200, 000 (Web high resolution images) | 10, 000 (images from user blur, occlusion etc.) |

How should we split this data-set for Train/Validation/Test

**Option1:** Mix all the data (210, 000) shuffle and split

|  |  |  |
| --- | --- | --- |
| Train | Validation | Test |
| 205, 000 samples | 2500 samples | 2500 samples |

This distribution make the Train/Validation/Test data-set distribution same but has HUGE disadvantage. In Validation set out of 2500 most of the samples are from Web images hence MODEL will be optimized for web images not for Mobile Camera images. Surely this is not what we want.

**Option2:** Split the images from Mobile Camera in 50, 25, 25 and distribute like below.

|  |  |  |
| --- | --- | --- |
| Train | Validation | Test |
| 205,000 [200,000(from web image) + 5K(mobile camera image)] | 2.5k(mobile camera) | 2.5k(mobile camera) |

Above data distribution will optimize MODEL for the mobile camera images or real test images. It is possible we may not get very high Train/Validation accuracy like *Option1* distribution but above kind of Data Split is good for LONG Term. *We can also try simple augmentation like FLIP on the 50% of the split to increase it’s ratio in TRAIN.*

**Repo/Code Directory Structure:** One can find a reference for a directory structure [here](https://github.com/cmawer/reproducible-model) and [here](https://blog.floydhub.com/structuring-and-planning-your-machine-learning-project/).

**Training A Model-I:** This stage of ML/DL project is most iterative. Below are some rule of Thumb which we can follow:

**Set up the end-to-end training/evaluation skeleton + get dumb baselines*. - Andrei Kaparthy***

1. Fix Random Seed: Use fix random seed until pipeline is not mature, it removes factor of variation and help reproducible.
2. Simplify: No fanciness at early stage, Example: No data augmentation
3. Human baseline: Whenever possible evaluate your own accuracy and compare to it. Take some random *k-samples* label twice one as ground truth and other as prediction.
4. Verify decreasing training loss
5. Picking the model: Don’t be a hero, find the most related paper and copy paste their simplest architecture that achieves good performance. Example if we are classifying images just copy paste a ResNet-50 for first run.
6. Optimizer: Adam is safe as start with start with learning rate = *3e-4* recommended by [Andrej Kaparthy](http://karpathy.github.io/2019/04/25/recipe/). *In my personal opinion use of RAdam is even better when Kaparthy suggested Adam, RAdam was not in picture.*
7. Pre-train: Start with pre-train model if available even though we have enough data.

**Training A Model -II:**

1. Introduce data augmentation slowly and record the performance for augmentation type. Good Survey on Augmentation [**URL**](https://link.springer.com/content/pdf/10.1186/s40537-019-0197-0.pdf).
2. Use of Cyclic LR, LR Recommendation etc.
3. Trying Layer wise Learning Rate. [[Ref1](https://github.com/keras-team/keras/issues/11934)] [[Ref2](https://blog.slavv.com/differential-learning-rates-59eff5209a4f)]
4. Experiment with different models.
5. Class-weights during training if you have highly imbalanced data problem ([weights can be computed by using sklearn](https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html)). One can also try [Oversampling and Under-sampling techniques.](https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis)
6. Do not rely on default momentum and decay rates TUNE them.
7. Document what worked and what didn’t.

**Model Analysis:** Bias error and Variance should be <1.5%. Up to <2% is acceptable.

**Bias Error:**

1. Train Bigger Model

2. Train longer with better optimizer

3. Tune for better Hyper-parameters.

**Variance:**

1. More data.

2. Regularization (L1/L2, Dropout,

Augmentation).

3. Tune/Search better Hyper-parameter.

**Human Error/**

**Proxy for Bayes Error**

**Bias Error(<1.5)**

**Train Error**

**Variance(<1.5)**

**Dev/Validation Error**

**Error Analysis:**

We are taking Cat vs Dog Image Classification task. Form a table like below from the Dev/Validation set.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Miss classified Images | Reason 1  (Dog) | Reason 2  (Great Cat) | Reason 3  (Blurry Image) | Incorrect Label  in Train Data-set | Comments |
| 1 |  |  |  |  | **Missed cat in background** |
| 2 |  |  |  |  | **Image blurry** |
| … |  |  |  |  | **Drawing of cat** |
| 100 |  |  |  |  | ... |
| % of Total | 8% | 43% | **61 %** | 4 % |  |

From the above table we can easily deduce that our first attempt should be to focus on ***Reason 3*** and so on.

**REFERENCES:**

1. Structuring Machine Learning Projects. - Andrew Ng [ [URL](https://www.coursera.org/learn/machine-learning-projects) ]
2. Organizing machine learning projects: project management guidelines. [ [URL](https://www.jeremyjordan.me/ml-projects-guide/) ]
3. How to plan and execute your ML and DL projects. [ [URL](https://blog.floydhub.com/structuring-and-planning-your-machine-learning-project/) ]
4. Things to Consider While Managing Machine Learning Projects. [ [URL](https://cloudxlab.com/blog/things-to-consider-while-managing-machine-learning-projects/) ]