PhD Progress: Development and application of deep learning methods for medical image analysis
Ανάπτυξη και υλοποίηση μεθοδολογιών βαθιάς μάθησης στην ανάλυση ιατρικής εικόνας

George Xenogiannopoulos

Biomedical Engineer

M.Sc. in Advanced Electronics Engineering

February 2021

1 System Development

Aiming to develop a tool that offers Machine Learning Operations (MLOps) with state-of-the-art (SOTA) algorithms, the work of the first year mainly focuses on building the infrastructure and developing a web application that serves the most common Machine Learning (ML) algorithms. We choose to build a web application in order to take advantage of the following:

- Cross platform compatibility: Building a web application using virtualization[1] or containerization[2] technologies, enables the deployment of the application to be Operating System (OS) agnostic.
- High accessibility: Easy access for many users through a web browser and network access.
- Minimum system requirements for the client: The minimum requirement for the client's machine is to be able to run a modern web browser.

The application's services can either be consumed from a local server or a remote server. In the second case the power of cloud computing can be another advantage. The application was build with containerization technology, using Docker[3], making it cross-platform. We choose Python as the main programming language which is one of the best programming languages for Artificial Intelligence (AI) and Web Development [1]. The project is being developed using open-source code and is hosted on a git repository in Bitbucket[4]. In more details, the web framework that we used is Django. It is written in Python and is one of the most popular frameworks following the MVC architectural pattern. MVC is an acronym for the words: Model, View, Control. The Model stands for the data structure and describes the database table's schema. The View part of the application is the one that visualizes the data to the user and in short, builds the User Interface (UI). Finally, the Control module is responsible to control the data flow and apply the business logic to the data. Without changing the essence of each module, Django renamed Control to View and View to Template resulting in the MVT architecture [2].

Django by default utilizes SQLite, which is also the default database for Python. In order to achieve better performance on a large scale and take advantage of features of both SQL (relational) and JSO(non-relational) querying, we chose PostgreSQL as our main database system.

Using Django and MVT architecture, we build a web platform named Medisp ML. Medisp ML uses Django and Django REST Framework (DRF) for Model and View or in other words for the Application Programming Interface (API) and a client side Single Page Application (SPA) for the UI, written in React. React is a JavaScript library developed by Facebook in order to build UIs using reusable components.

Medisp ML was built to carry ML tasks which usually are time and resource consuming. Since Medisp ML runs on a web server, a long-running ML task could occupy the server making it incapable for receiving

[1] `https://opensource.com/resources/virtualization`
[2] `https://www.ibm.com/cloud/learn/containerization`
[3] `https://www.docker.com/`
[4] `https://bitbucket.org/`

new request. The way we decided to sort this out was to use a Celery worker to run the tasks and Redis[5] as message broker to distribute tasks. When a request for heavy task reaches the server, then the server sends a message to the broker and the broker appends the message to a job queue. The message contains all the information that a worker needs to fulfill a job. It is the broker's responsibility to find non-busy workers and distribute the pending jobs until the end of the queue.

Finally, for the big variety of Machine Learning and Deep Learning libraries that Python offers, we picked the most famous and widely used, scikit-learn[6] and TensorFlow[7] in order to build out operations.

2 Machine Learning Algorithms

The ML operations that we build aim to solve classification problems. For our operations we used the following classifiers, feature reduction methods as well as evaluation techniques. We built a discriminant analysis pipeline which examines which features of a given dataset are the most significant to be used on a pattern recognition system. The pipeline can run on any given combination of the following classifiers, feature reduction methods and evaluation techniques. All the methods are fully customizable with parameters. In our implementation we tried to integrate, wherever possible, ML methods directly from the scikit-learn[6] library.

Classifiers

1. Minimum Distance Classifier (MDC)
2. k-nearest Neighbors (KNN)
3. Bayesian (BAYESIAN)
4. Linear Discriminant Analysis (LDA)
5. Logistic Regression (LOGREG)
6. Perceptron (PERCEPTRON)
7. Multi-layer Perceptron (MLP)
8. Support Vector Machine (SVM)
9. Random Forest (RANDOMFOREST)
10. Decision Tree (DECISIONTREE)
11. Extra Trees (EXTRATREES)

Feature Reduction

1. Correlation Ranking (CORRELATION)
2. Principal Component Analysis (PCA)
3. Recursive Feature Elimination (RFE)

System Evaluation

1. Leave-One-Out (LOO)
2. Hold Out (HOLDOUT)
3. K-Fold (KFOLD)
4. Bootstrap (BOOTSTRAP)

[5] `https://redis.io/`
[6] `https://scikit-learn.org`
[7] `https://www.tensorflow.org`

3 Results

The algorithms and ML operations that we developed were tested using the following dataset: Image Analysis features extracted from CRC (Colorectal Cancer) histologic images. The CRC dataset contains 69 features belonging to 2 classes (High Grade, Low Grade) with 26 and 28 samples respectively. The following table summarizes the results achieved by seven classifiers, using the K-Fold evaluation method, the RFE feature reduction method and maximum four features combination.

Classifier	Accuracy	Sensitivity	Specificity
KNN	0.79	0.79	0.8
SVM	0.70	0.83	0.63
DECISIONTREE	0.83	0.82	0.84
RANDOMFOREST	0.81	0.82	0.80
EXTRATREES	0.79	0.79	0.8
LDA	0.74	1.0	0.65
BAYESIAN	0.66	0.91	0.59

References

[1] A. Singh and S. Paul. *Hands-On Python Deep Learning for the Web: Integrating neural network architectures to build smart web apps with Flask, Django, and TensorFlow*. Packt Publishing, 2020.

[2] S. Holzner. *Django: Visual QuickPro Guide*. Visual QuickPro Guide. Pearson Education, 2009.