Machine Learning Report Project

FOZAME ENDEZOUMOU Armand Bryan[PGE4], Maheni SOUMAH[PGE3], Jessica MBOUNKAP[PGE3], Habiba DJIGO[PGE3], Darryl TOWA[PGE3], LEUMALEU MBOUYOM Arnold[PGE3]

Abstract

This academic project, supervised by Dr. Hanoune, aimed to master fundamental concepts in machine learning. It involved completing four case studies, along with a few additional tasks. The first project focused on classification and regression, the second on image classification, the third on unsupervised learning, and the last on object detection and segmentation. Additionally, participants were tasked with implementing classification and regression using AdaBoost, as well as developing a convolution algorithm from scratch, including stride and padding. These projects were completed and documented on our GitHub repository, which can be found at the following link: `https://github.com/Bryan-Foxy/ml-project`. In the remainder of this report, we will explain our choice of algorithms used and present the results obtained.

1 Introduction

To successfully carry out our projects, the first critical step was to carefully select the datasets we would work on. After thorough consideration, we chose a variety of datasets, each targeting a specific aspect of machine learning. These datasets are stored in a shared Drive folder, accessible via the following link. Here is an overview of the selected datasets that will form the foundation of our work:

- For classification/regression : We opted for the renowned Wine dataset, which offers a fertile ground for various predictive analyses.
- In image classification : We focused on images from American Sign Language, a choice motivated by the desire to make technology more inclusive.
- For unsupervised learning : We chose the dataset of cancers in Lake County, Illinois, USA, to explore models capable of identifying patterns and anomalies.
- Regarding image segmentation : Our attention turned to images and masks of lungs affected by Covid-19, a timely and critically important subject for public health.

To facilitate collaboration and share our progress, we created a GitHub repository, a central space where our work can be versioned and reviewed efficiently by all team members. Additionally, to leverage advanced computational resources, we chose to use Google Colab, which allows us access to powerful GPUs, significantly speeding up the processing time of our analyses.

This infrastructure enabled us to dive into exploring these datasets with all the necessary resources at our disposal. Our goal is not only to extract relevant insights from these data but also to develop robust and efficient machine learning models. Each project, targeting a different aspect of machine learning, is designed to deepen our theoretical understanding while confronting us with practical challenges, thus preparing us to tackle real-world issues.

2 Classification/Regression

In our inaugural project, we embarked on both classification and regression analyses using the Wine dataset. A key aspect of this project was not only to analyze this dataset but also to integrate it with another dataset from the same domain, albeit with differing columns. This task required a meticulous approach to data preparation and analysis.

During our initial analysis, we identified missing data within our datasets. To address this issue, we employed the Newton Interpolation method, a strategic choice for estimating missing values by leveraging nearby data points. The Newton Interpolation formula we applied is as follows:

$$P_n(x) = \sum_{i=0}^{n} \beta_i w_i(x) with w_i(x) = \prod_{j=0}^{i-1} (x - x_j), \forall i \in [1, .., n] and w_0(x) = 1$$

This equation succinctly captures the essence of Newton's Interpolation, where $P_n(x)$ represents the polynomial of degree n approximating the function. Each β_i is a coefficient determined through divided differences based on the known data points, and $w_i(x)$ are the basis polynomials constructed from the input data points x_j.

After addressing the missing data and thoroughly analyzing the datasets, we proceeded with our experiments, which encompassed four distinct analyses: two focused on classification/regression with a single dataset, and two more utilizing the merged dataset. For these tasks, we chose the XGBoost model, renowned for its effectiveness and efficiency across various machine learning challenges. These are the results:

Table 1: Table of accuracy of the both classification model

Models	Accuracy
$\text{Model}_{classification}$	99.23
$\text{Model}_{classification} assembly$	99.54

Table 2: Table of accuracy of the both regression model

Models	MSE	RMSE	MAE
$\text{Model}_{regression}$	0.39	0.63	0.45
$\text{Model}_{regression} assembly$	0.41	0.64	0.49

The use of XGBoost was pivotal, given its proven track record in handling both classification and regression tasks with high accuracy. This model's capacity to manage sparse data and its versatility in dealing with various types of predictive modeling tasks made it an ideal choice for our project.

3 Sign Language Classification

Our classification project involved the use of a sign language dataset in order to extract the correct translation. To do this, we mainly used Pytorch modules, which enabled us to study the structure of the dataset, to access optimizers and various transformations required for optimum results, such as normalization and restructuring. The data separation required for proper model training was carried out using a function that enabled us to divide the data into documents containing 80for training and testing respectively. After image augmentation and normalization -to rotate, reverse and add brightness- with torchvision, we used a data loader to facilitate downloading, iterations and limit certain modifications that could affect model accuracy by allow data to be divided into batches and processed using a GPU. The Figure 1 show the result: Dealing with a multiclass classification

Figure 1: Image after transformation in the DataLoader

and a database with few images, we tested different models, namely : Alexnet, VGGNet, ResNet-150... the obvious choice was the Inception model thanks to the satisfactory results it offered. The Inception model, commonly referred to as GoogleNet, was introduced by Christian Szegedy et al. in their paper titled "Going Deeper with Convolutions" [3]. The uniqueness of Inception lies in its ability to perform convolutions, pooling, etc. simultaneously, significantly reducing the number of parameters while maintaining effectiveness, as illustrated in Figure 2. The Figure 3 explain well:

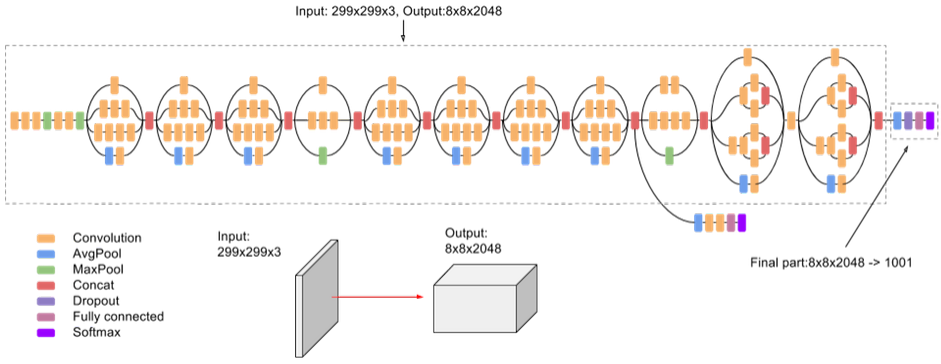

Figure 2: Inception

Comparison					
Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Figure 3: Image after transformation in the DataLoader

It have also a limited number of parameters, especially:
logits(the parameter controls whether to include an auxiliary classifier, for this case, we decined to not include an auxiliary classifier),SGD (with to Momentum, to stabilize the learning), making it easier to train.
The evaluation of our model has enabled us to congratulate its highly satisfactory performance in terms of sensitivity and prediction. Let's see the loss and the confusion matrix Figure 4:

(a) Loss Accuracy

(b) Confusion Matrix

Figure 4: Evaluation

Finally, we deployed our model by playing the game, inserting our own data and mimicking different signs. We had 5 images, representing 5 different signs made by 5 different people, and the models correctly predicted 4 out of 5 images. We quickly realized that the "C" sign was very similar to the "R" sign in the training database, and that this was a factor that pushed our model away from faultlessness.Nevertheless, generally speaking, the model obtained is reliable, precise and sensitive.

4 Unsupervised learning

In this section, we have applied our unsupervised learning skills with the goal of visualizing the different cancer regions across Lake County. Additionally, we aim to understand how dimensionality reduction works by implementing it.

Initially, we are working with two distinct types of data:

- A .geopandas file that contains the geographical coordinates of Lake County.
- A .csv file that includes data on cancer cases found in Lake County.

To display the geographical file, we utilized `geopandas` for mapping visualization. After preprocessing the .csv data, we determined the number of clusters using the elbow method and then applied KMEANS to identify distinct regions. These regions are then overlaid on our map. Also to find the best k, we have perform the famous elbow method and the Figure 5 is the results.

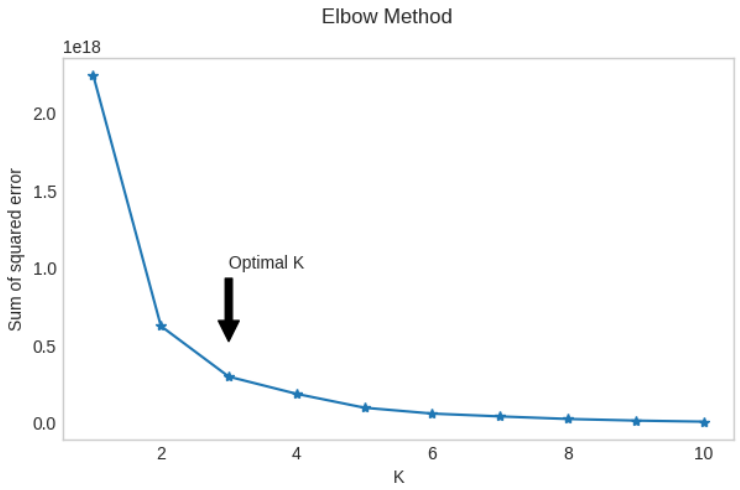

Figure 5: Optimal K

Next, we move on to dimensionality reduction to determine the number of components necessary to retain 95 percent or more of the information. This technique is employed for feature selection as some columns may contain white noise without contributing any useful information. We visualize the data using PCA and t-SNE in 2D and employ PCA in 3D, which allows us to clearly see the clusters identified earlier with KMEANS.

The results are in the Figure 6

We can observe the presence of the 3 cancer regions in this county.

5 Segmentation Covid Lunges

This study aims to directly support medical practitioners and researchers in the health sector. The images are X-rays of individuals affected by Covid-19, and to assist those focused on medical research, we have sought to highlight and analyze the lungs while discarding other irrelevant information. To achieve this, we implemented a semantic segmentation algorithm to segment the lungs from the images.
We undertook preparatory steps that involved creating a CustomDataset, splitting the data (80-10-10), and then a DataLoader to process the data in mini-batches and compute in parallel. The

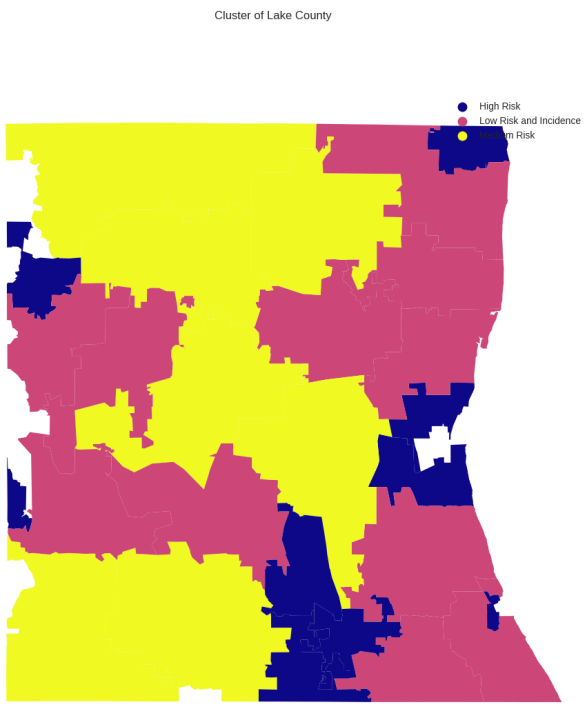

Figure 6: Cluster Lake Comty

following figure 7 provides a glimpse of the final images contained in the DataLoader, ready to be fed into the model:

Following this, we developed the model. We utilized the U-Net architecture, a deep convolutional neural network architecture introduced by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in their 2015 paper titled *U-Net: Convolutional Networks for Biomedical Image Segmentation* [2]. This architecture is specifically designed for the segmentation of biomedical images and is characterized by its U-shaped structure, allowing for precise localization and efficient data usage. Thanks to its copy-and-crop mechanism, U-Net is capable of capturing the context of images at different resolution levels, making it a preferred choice for our study. The U-Net architecture is illustrated in the following figure 8:

However, we used `LeakyReLU` as the activation function because with `ReLU`, we often encountered issues with dead neurons and always positive gradients. As an optimizer, we chose `Adam` with a learning rate of 0.001, which optimizes as we use an LR Scheduler, and a loss function `DiceLoss`. We attempted using both `Xavier Initialization` and `He Initialization` to adjust the model weights for better convergence, but this did not yield good results, so it is preferable to keep the weights initialized randomly. Additionally, to further enhance our approach, we combined the strength of the U-Net architecture with transformers to create an attentive U-Net. This innovative approach leverages the global attention mechanisms introduced by Vaswani et al. [4] in their groundbreaking work on Transformers, as well as the successful adaptation of these principles to computer vision with Vision Transformers (ViT) by Dosovitskiy et al. [1].

For the results, with early stopping to prevent overfitting, U-Net underwent 29 epochs while U-Net with Attention only had 10 epochs. The following figure 9 shows the training of both models.

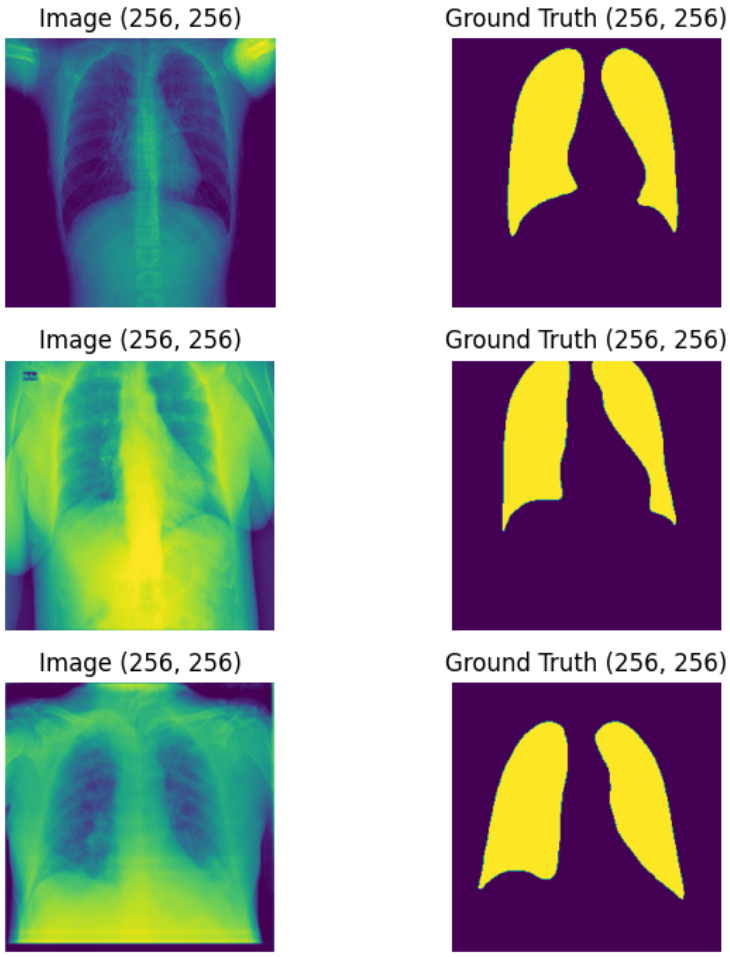

Figure 7: Lung images in the DataLoader

After computing the metrics, the results are summarized in the table below 3:

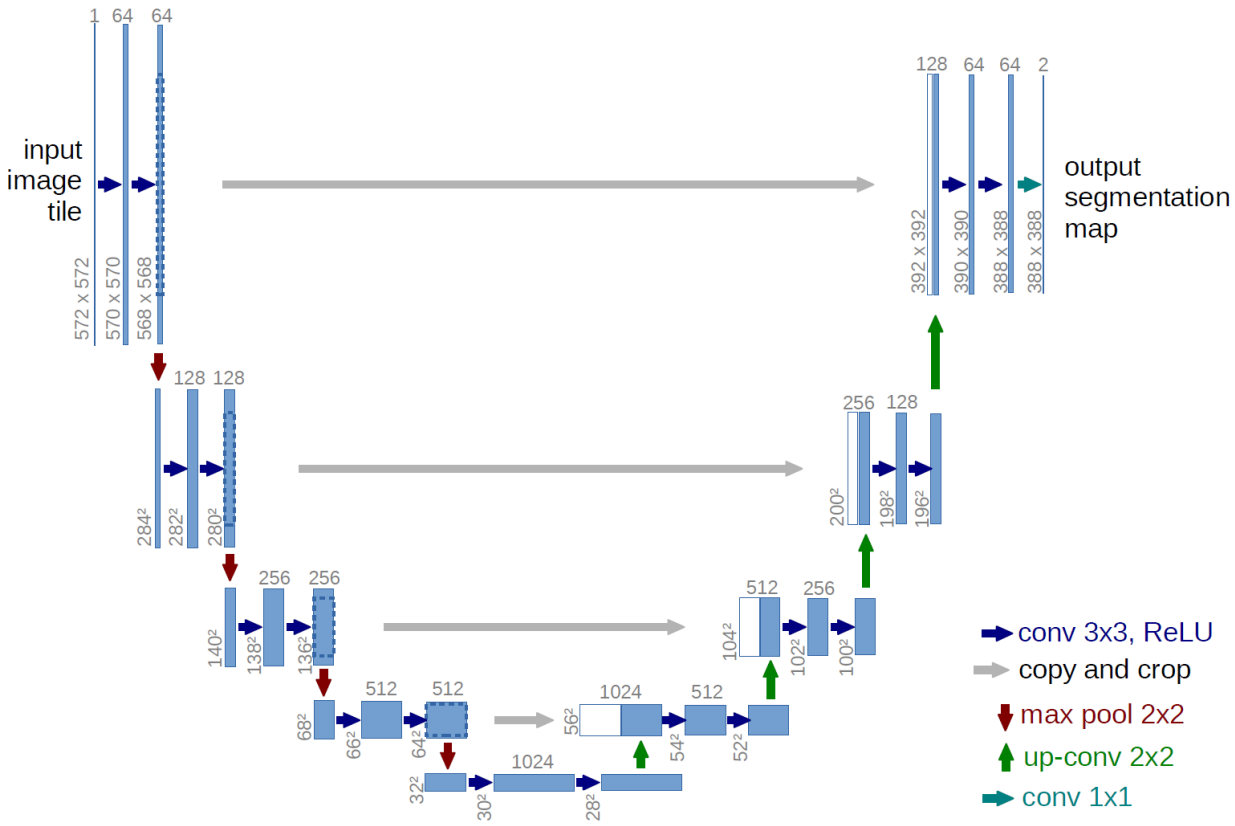

Figure 8: U-Net architecture

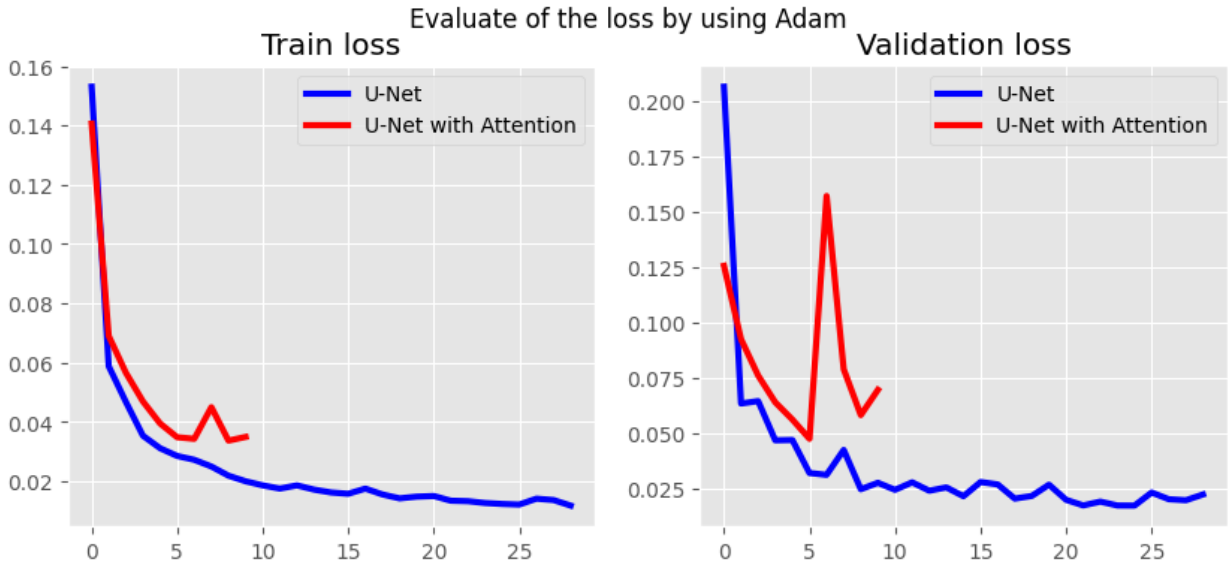

Figure 9: Loss (U-Net U-Net with Attention)

Table 3: Performance comparison between U-Net with Attention and classic U-Net

(a) U-Net with Attention

Metric	**Value**
Accuracy	0.9513
IoU	0.8068
F1 Score	0.8931
Precision	0.9535
Recall	0.8398

(b) Classic U-Net

Metric	**Value**
Accuracy	0.9874
IoU	0.9488
F1 Score	0.9737
Precision	0.9861
Recall	0.9616

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, 2015.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.