Mohamed V University

École Nationale Supérieure d'Informatique et d'Analyse des Systèmes

Deep Learning Project

Truss Structure Performance Prediction

Authors
Fatiha BARRADE
Fatima NOUTFI

Supervisors
Pr. CHIHEB Raddouane
Mr. MOUDNI Ahmad
Mr. OUHAMOUDDO Said

June 13, 2024

Abstract

This project employs regression techniques to predict weight and energy values for truss structures and Convolutional Neural Networks (CNNs) for stability classification, aiming to provide engineers with valuable insights into structural characteristics and safety considerations. By rigorously preprocessing data and designing appropriate models, the project strives to accurately predict weight and energy values, aiding in structural optimization and performance evaluation. Additionally, the utilization of CNNs enables the classification of truss structures based on stability, contributing to enhanced safety and structural integrity in civil engineering applications. Through the integration of machine learning techniques, this project seeks to advance the field by offering efficient tools for structural analysis and decision-making processes.
Keywords : Truss structures, regression, Multilayer Perceptron, classification, Convolutional Neural Networks, image preprocessing, structural stability.

Contents

1 Introduction

Truss structures are essential components in the field of civil and structural engineering, prominently used in the construction of bridges, telecommunications towers, stadium roofs, and other critical infrastructures. These structures, composed of interconnected beams and bars, are designed to efficiently support and distribute loads, leveraging triangulated configurations to maximize stability and structural integrity.

The optimization of truss structures is a complex and critical task, involving the design of trusses that meet specific performance criteria while minimizing objectives such as the total weight, material stresses, and costs associated with fabrication and installation. This process is essential for ensuring that structures are not only safe and durable but also cost-effective and efficient in terms of material usage.

Traditionally, the design of truss structures has relied on analytical and empirical methods. However, the advent of advanced computational techniques has revolutionized this field, enabling more sophisticated and precise optimization. Heuristic optimization methods, such as genetic algorithms and simulated annealing, have proven particularly effective in exploring the vast design space of truss configurations and identifying optimal solutions under various constraints.

The goal of this project is to harness these advanced optimization techniques and integrate them with machine learning methods to develop a robust framework for the design and analysis of truss structures. By utilizing a comprehensive dataset of truss designs and employing predictive modeling, we aim to enhance the ability to predict critical performance metrics and classify the stability of these structures.

This framework should address the following objectives:

- 1- Prediction of Weight and Energy: Propose neural network architectures capable of accurately predicting the weight and deformation energy of truss structures based on design parameters and constraints.
- 2- Classification of Truss Structures: Design and implement a convolutional neural network (CNN) architecture to classify truss structure images based on stability and compliance with design constraints.

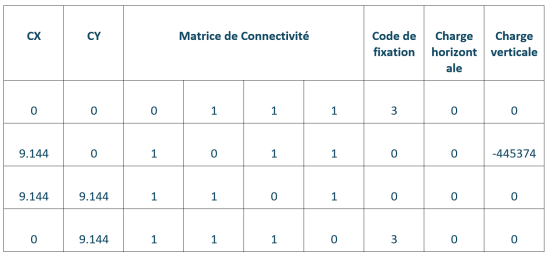
2 Dataset

2.0.1 Truss Structure Data

The necessary data for defining a truss structure encompasses several fundamental elements:

1. **Geometry:** The dimensions of the structure, including length, width, and height, as well as the arrangement of nodes and truss members.
2. **Materials:** Properties of the materials used for the truss members, such as tensile strength, compressive strength, stiffness, etc.
3. **Loads:** Applied forces on the structure, such as wind loads, snow loads, traffic loads, etc. These loads can be either static or dynamic.
4. **Constraints and Design Criteria:** Maximum allowable constraints on the truss members, permissible deformations, safety requirements, building codes compliance, etc.
5. **Boundary Conditions:** Boundary conditions such as fixed supports, joints, anchoring points, etc.
6. **Optimization Objectives:** Goals to achieve, such as minimizing the total weight of the structure, maximizing stability, minimizing deformations, etc.

The data structure for truss structures is typically organized in matrix form, with a fixation code specifying the boundary conditions. Here's an example of a data structure:



CX	CY	Matrice de Connectivité				Code de fixation	Charge horizontale	Charge verticale
0	0	0	1	1	1	3	0	0
9.144	0	1	0	1	1	0	0	-445374
9.144	9.144	1	1	0	1	0	0	0
0	9.144	1	1	1	0	3	0	0

Figure 2.1: Truss Structure

This data structure contains crucial information for modeling and analyzing truss structures, providing a clear representation of geometry, materials, loads, constraints, and optimization objectives.

2.1 Data Description

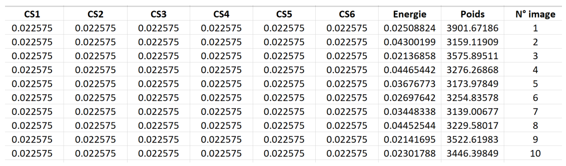
We utilized a variety of datasets for the truss structure project. Here's a breakdown of each dataset and its components:

2.1.1 Structure Dataset

The structure dataset contains information about the truss structure's characteristics and performance metrics. It includes features such as:

- CS1, CS2, CS3, CS4, CS5, CS6: Coefficients representing certain properties of the structure.
- Energie: Energy associated with the structure.
- Poids: Weight of the structure.
- N° image: Image number corresponding to the generated structure.

This dataset serves as the primary source of data for training and evaluating predictive models.

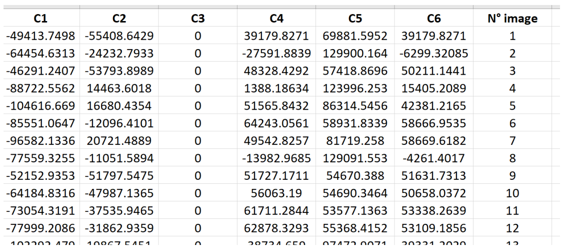


CS1	CS2	CS3	CS4	CS5	CS6	Energie	Poids	N° image
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.02508824	3901.67186	1
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.04300199	3159.11909	2
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.02136858	3575.89511	3
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.04465442	3276.26868	4
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.03676773	3173.97849	5
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.02697642	3254.83578	6
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.03448338	3139.00677	7
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.04452544	3229.58017	8
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.02141695	3522.61983	9
0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.02301788	3446.39849	10

Figure 2.2: Truss Structure Dataset

2.1.2 Constraint Dataset

The constraint dataset provides details about the constraints applied to the truss structure. It includes:

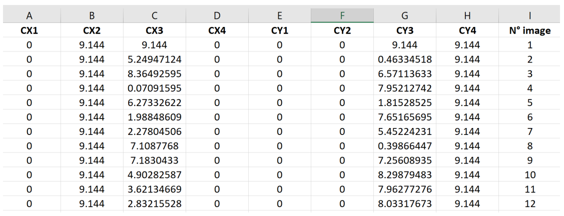


C1	C2	C3	C4	C5	C6	N° image
-49413.7498	-55408.6429	0	39179.8271	69881.5952	39179.8271	1
-64454.6313	-24232.7933	0	-27591.8839	129900.164	-6299.32085	2
-46291.2407	-53793.8989	0	48328.4292	57418.8696	50211.1441	3
-88722.5562	14463.6018	0	1388.18634	123996.253	15405.2089	4
-104616.669	16680.4354	0	51565.8432	86314.5456	42381.2165	5
-85551.0647	-12096.4101	0	64243.0561	58931.8339	58666.9535	6
-96582.1336	20721.4889	0	49542.8257	81719.258	58669.6182	7
-77559.3255	-11051.5894	0	-13982.9685	129091.553	-4261.4017	8
-52152.9353	-51797.5475	0	51727.1711	54670.388	51631.7313	9
-64184.8316	-47987.1365	0	56063.19	54690.3464	50658.0372	10
-73054.3191	-37535.9465	0	61711.2844	53577.1363	53338.2639	11
-77999.2086	-31862.9359	0	62878.3293	55368.4152	53109.1856	12
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 2.3: Constraints Dataset

2.1.3 Aplatit Dataset

The aplatit dataset contains information related to the flattened structure.

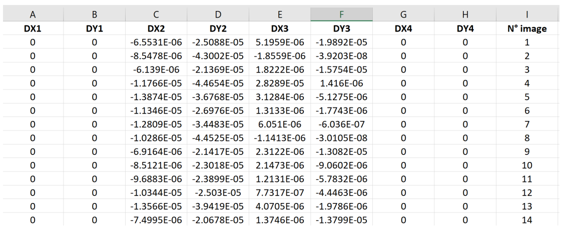


A	B	C	D	E	F	G	H	I
CX1	**CX2**	**CX3**	**CX4**	**CY1**	**CY2**	**CY3**	**CY4**	**N° image**
0	9.144	9.144	0	0	0	9.144	9.144	1
0	9.144	5.24947124	0	0	0	0.46334518	9.144	2
0	9.144	8.36492595	0	0	0	6.57113633	9.144	3
0	9.144	0.07091595	0	0	0	7.95212742	9.144	4
0	9.144	6.27332622	0	0	0	1.81528525	9.144	5
0	9.144	1.98848609	0	0	0	7.65165695	9.144	6
0	9.144	2.27804506	0	0	0	5.45224231	9.144	7
0	9.144	7.1087768	0	0	0	0.39866447	9.144	8
0	9.144	7.1830433	0	0	0	7.25608935	9.144	9
0	9.144	4.90282587	0	0	0	8.29879483	9.144	10
0	9.144	3.62134669	0	0	0	7.96277276	9.144	11
0	9.144	2.83215528	0	0	0	8.03317673	9.144	12

Figure 2.4: Flattenet Structure Dataset

2.1.4 Déplacement Dataset

The déplacement dataset contains data regarding the displacement of the structure.



A	B	C	D	E	F	G	H	I
DX1	**DY1**	**DX2**	**DY2**	**DX3**	**DY3**	**DX4**	**DY4**	**N° image**
0	0	-6.5531E-06	-2.5088E-05	5.1959E-06	-1.9892E-05	0	0	1
0	0	-8.5478E-06	-4.3002E-05	-1.8559E-06	-3.9203E-08	0	0	2
0	0	-6.139E-06	-2.1369E-05	1.8222E-06	-1.5754E-05	0	0	3
0	0	-1.1766E-05	-4.4654E-05	2.8289E-05	1.416E-06	0	0	4
0	0	-1.3874E-05	-3.6768E-05	3.1284E-06	-5.1275E-06	0	0	5
0	0	-1.1346E-05	-2.6976E-05	1.3133E-06	-1.7743E-06	0	0	6
0	0	-1.2809E-05	-3.4483E-05	6.051E-06	-6.036E-07	0	0	7
0	0	-1.0286E-05	-4.4525E-05	-1.1413E-06	-3.0105E-08	0	0	8
0	0	-6.9164E-06	-2.1417E-05	2.3122E-06	-1.3082E-05	0	0	9
0	0	-8.5121E-06	-2.3018E-05	2.1473E-06	-9.0602E-06	0	0	10
0	0	-9.6883E-06	-2.3899E-05	1.2131E-06	-5.7832E-06	0	0	11
0	0	-1.0344E-05	-2.503E-05	7.7317E-07	-4.4463E-06	0	0	12
0	0	-1.3566E-05	-3.9419E-05	4.0705E-06	-1.9786E-06	0	0	13
0	0	-7.4995E-06	-2.0678E-05	1.3746E-06	-1.3799E-05	0	0	14

Figure 2.5: Displacement dataset

2.1.5 Image Dataset

The image dataset contains 2D images of each truss structure. There are a total of 2001 images in this dataset, with each image corresponding to a specific truss structure in the project. These images serve as visual representations of the geometry and configuration of the truss structures, providing valuable insights for analysis and interpretation.

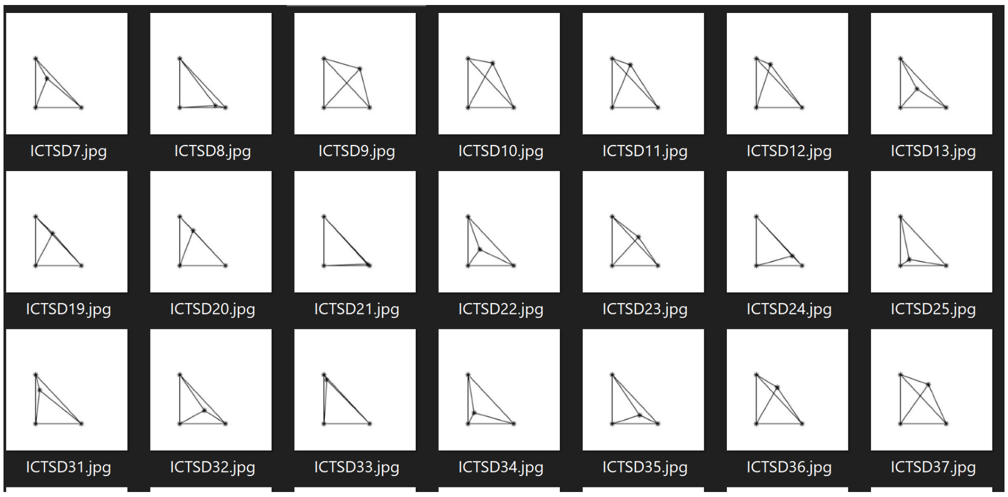


Figure 2.6: Truss Structure Images

3 Implementation

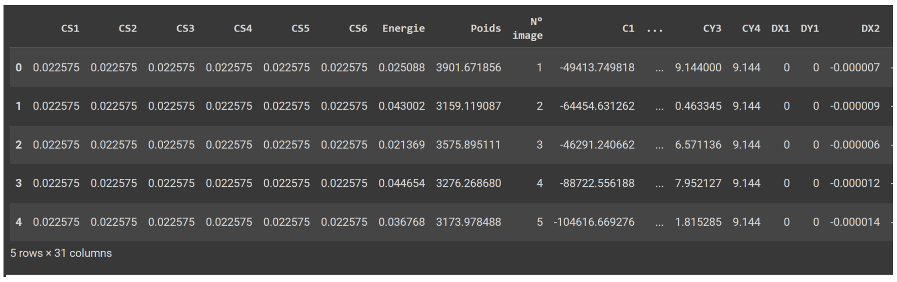
3.1 Weight and Energy Prediction

In this section, we will embark on the first task of the project, which is regression. Our objective is to predict the weight and energy of a truss structure given its information using a Multi-Layer Perceptron (MLP).

3.1.1 Data Preparation

The first step involves preparing the dataset. The data consists of various features describing the truss structure. Initially, we utilized the Truss Structure dataset, but we noticed that it contains duplicate information. To enhance the dataset, we concatenated several datasets: Truss Structure, Flattened, Constraints, and Displacement. These combined features provide a comprehensive representation of the truss structures and are transformed into a suitable format for the MLP. The following preprocessing steps are performed:

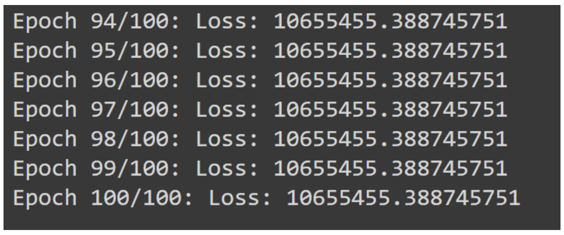
- **Removing Irrelevant Features**: Columns that have the same value for all rows or contain only zeros are removed as they do not contribute to the learning process.
- **Normalization**: To ensure that all features contribute equally to the learning process, we normalize the input data. For the target variable 'weight', which has large values, normalization is also applied to prevent the loss from exploding. For 'energy', since the values are small, normalization is not applied. The scaler used for 'weight' is saved for inverse transformation during prediction.



	CS1	CS2	CS3	CS4	CS5	CS6	Energie	Poids	N° image	C1	...	CY3	CY4	DX1	DY1	DX2
0	0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.025088	3901.671856	1	-49413.749818	...	9.144000	9.144	0	0	-0.000007
1	0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.043002	3159.119087	2	-64454.631262	...	0.463345	9.144	0	0	-0.000009
2	0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.021369	3575.895111	3	-46291.240662	...	6.571136	9.144	0	0	-0.000006
3	0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.044654	3276.268680	4	-88722.556188	...	7.952127	9.144	0	0	-0.000012
4	0.022575	0.022575	0.022575	0.022575	0.022575	0.022575	0.036768	3173.978488	5	-104616.669276	...	1.815285	9.144	0	0	-0.000014

5 rows × 31 columns

Figure 3.1: New Dataset after merging



```
Epoch 94/100: Loss: 10655455.388745751
Epoch 95/100: Loss: 10655455.388745751
Epoch 96/100: Loss: 10655455.388745751
Epoch 97/100: Loss: 10655455.388745751
Epoch 98/100: Loss: 10655455.388745751
Epoch 99/100: Loss: 10655455.388745751
Epoch 100/100: Loss: 10655455.388745751
```

Figure 3.2: Loss Explosion before target Normalization

- **Splitting**: The dataset is split into three parts:
  - **Training Set**: Used to train the model.(80% of the dataset)
  - **Validation Set**: Used to tune the model's hyperparameters and prevent overfitting. (25% Of the train dataset)
  - **Test Set**: Used to evaluate the final model performance. (0.2 % of the dataset)

3.1.2 Model Architecture

The model architecture consists of a Multilayer Perceptron (MLP) designed for regression tasks. It comprises three fully connected layers with ReLU activation functions between them. The input layer has 13 units corresponding to the features of the input data, while the output layer has a single unit, as it is a regression task. The hidden layers consist of 64 and 32 units, respectively. ReLU activation functions are applied after each linear transformation to introduce non-linearity into the model.

For training, Mean Squared Error (MSE) loss is utilized as the loss function, which measures the average squared difference between the predicted and true values. The Adam optimizer is employed to update the model parameters during training, which efficiently adapts learning rates for each parameter based on the past gradients.

This architecture aims to learn complex relationships between the input features and the target output for regression tasks, allowing the model to predict continuous values accurately.

3.1.3 Training and Results

Training

The training process for the weight prediction model and the energy prediction model has yielded impressive results. For the weight prediction model, the training loss and mean absolute error (MAE) are notably low, with values of 0.000124 and 0.0082, respectively. Similarly, the validation loss and MAE, though slightly higher at 0.00123 and 0.0273,

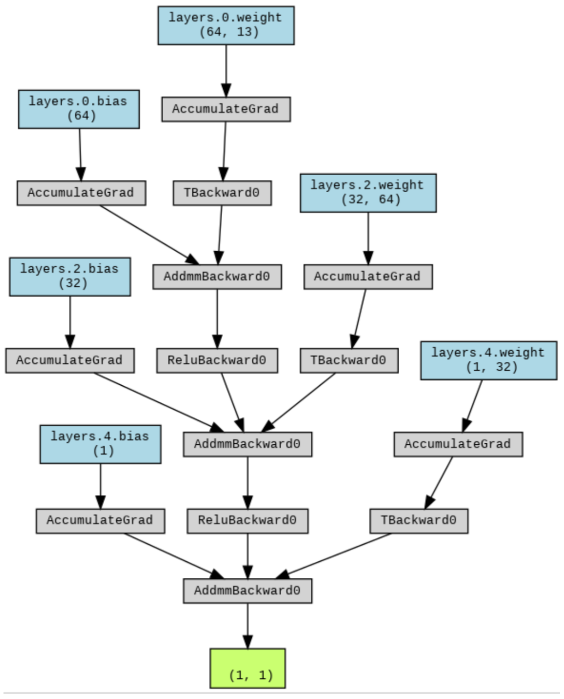


Figure 3.3: Model Architecture

demonstrate robust performance on unseen data. This suggests that the model has learned the underlying patterns in the data well and generalizes effectively. On the other hand, the energy prediction model has achieved even more remarkable results, with an exceedingly low training loss of 0.000001 and a training MAE of 0.000708. The validation loss and MAE are also impressively low at 0.000015 and 0.003396, respectively. Such minimal errors indicate that the model has captured the intricacies of the energy data with remarkable precision. Overall, both models exhibit strong performance, demonstrating their efficacy in their respective prediction tasks. These results speak to the effectiveness of the training process and the potential utility of the models in practical applications.

3.1.4 Test Results

The test results for both the weight prediction and energy prediction models further validate their robustness and effectiveness. In the case of the weight prediction model, the test mean absolute error (MAE) is slightly higher at 0.0426, indicating a small deviation from the actual weight values. However, the test mean squared error (MSE) and root mean squared error (RMSE) are still relatively low at 0.00330 and 0.0575, respectively. Moreover, the test coefficient of determination (R^2)$valueof0.9967signifiesanexceptionallevelofaccuracy, withthemod$

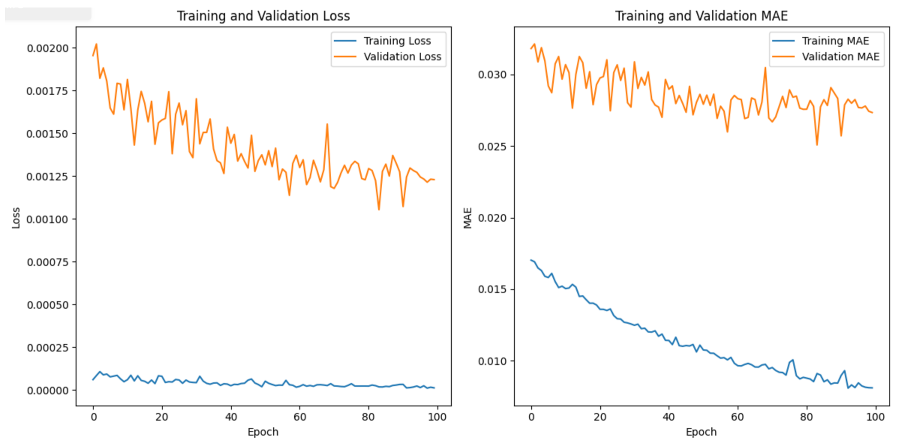


Figure 3.4: Train and Validation MSE and MAE for Weight Prediction

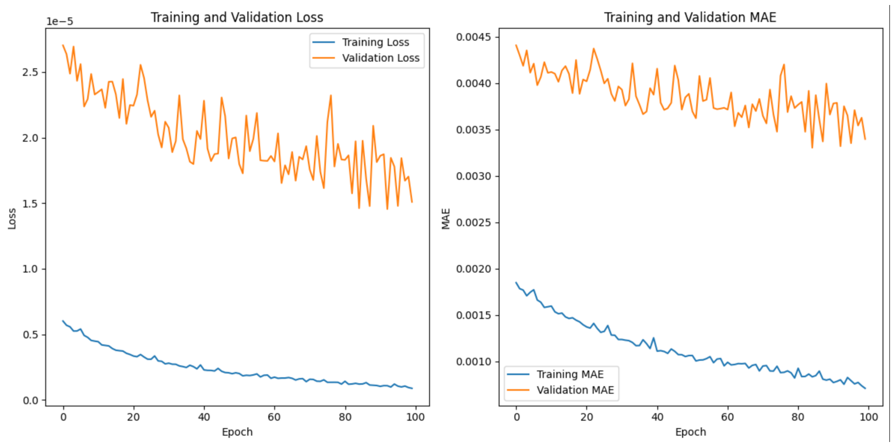


Figure 3.5: Train and Validation MSE and MAE for Energy Prediction

3.2 Genetic Algorithm For Neurons number choice

In order to choose the optimal number of neurons for predicting the weight and energy of truss structures, we used a Genetic Algorithm (GA). This is a combinatorial optimization problem that can be formulated as follows:

$$\text{Maximize} \quad f(x) \tag{3.1}$$

$$\text{Subject to} \quad \sum_{i=1}^{N} x_i \leq M \tag{3.2}$$

$$x_i \in \{0,1\} \quad \forall i \in N \tag{3.3}$$

where:

- N is the total number of neurons,
- M is the maximum allowed number of neurons,
- x_i is a binary decision variable indicating the inclusion or exclusion of the i-th neuron,
- $f(x)$ is the fitness function, defined for 0/1 loss as:

$$f(x) = \frac{1}{\sum_{i=1}^{N} x_i \cdot I(y_i \neq \hat{y}_i) + 1} \tag{3.4}$$

Here, $I(y_i \neq \hat{y}_i)$ is an indicator function that equals 1 if the true label (y_i) is not equal to the predicted label $(\hat{y}_i)$, and 0 otherwise.

3.2.1 Genetic Algorithm Approach

Definition

Genetic Algorithm (GA) is a population-based global search optimization technique rooted in the principles of natural selection and genetic inheritance. It maintains a population of potential solutions that evolve over successive generations through genetic operations such as crossover and mutation, enhancing the search for optimal or near-optimal solutions.

Steps of Genetic Algorithm

- **Initialization**: Create an initial population comprising several individuals, each representing a potential solution with a binary sequence indicating the presence or absence of neurons.
- **Fitness Evaluation**: Assign a fitness value to each individual based on a chosen fitness function, which quantifies the solution's quality.
- **Selection**: Choose individuals for the mating pool based on their fitness, promoting the inheritance of favorable traits.
- **Reproduction (Generating a New Population)**: Select pairs of parents from the mating pool to produce offspring.
- **Crossover (Recombination)**: Apply crossover to pairs of parents to create offspring, promoting genetic diversity.
- **Mutation**: Introduce randomness by mutating offspring, enhancing diversity within the population.

- **Replacement**: Replace the old population with the newly generated population of offspring, repeating the process until a stopping criterion is met.

3.2.2 Coding Approach

We employ a binary encoding strategy to represent the configuration of neurons in a neural network. Each bit in the binary sequence corresponds to the presence ('1') or absence ('0') of a specific neuron. This encoding method offers a concise and computationally efficient way to represent diverse neuron configurations, ensuring a balance between adequate exploration of the search space and computational efficiency.

3.2.3 Fitness Function

Selecting an appropriate fitness function is crucial for guiding the genetic algorithm toward optimal neuron configurations. Three potential fitness functions were considered: 0/1 loss inverse, accuracy, and F1 score. Ultimately, we chose Mean Squared Error (MSE) as our fitness metric. Expressed as a negative value to align with the maximization objective of GA, the fitness function is defined as:

$$\text{Fitness} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

where y_i and $\hat{y}_i$ are the true and predicted values, respectively. This formulation enables the genetic algorithm to find neural network configurations that minimize the MSE, thereby enhancing the accuracy of weight and energy predictions for truss structures.

3.2.4 Results

We meticulously explored and experimented with different sets of hyperparameters for the Genetic Algorithm (GA) until we attained the most precise and effective results.

The application of the genetic algorithm to optimize the number of neurons yielded promising results. For the energy prediction model, the Mean Squared Error (MSE) achieved was 0.00013. For the weight prediction model, the MSE was 0.928922. These results demonstrate the effectiveness of using genetic algorithms to optimize neural network configurations for predicting structural properties of truss systems.

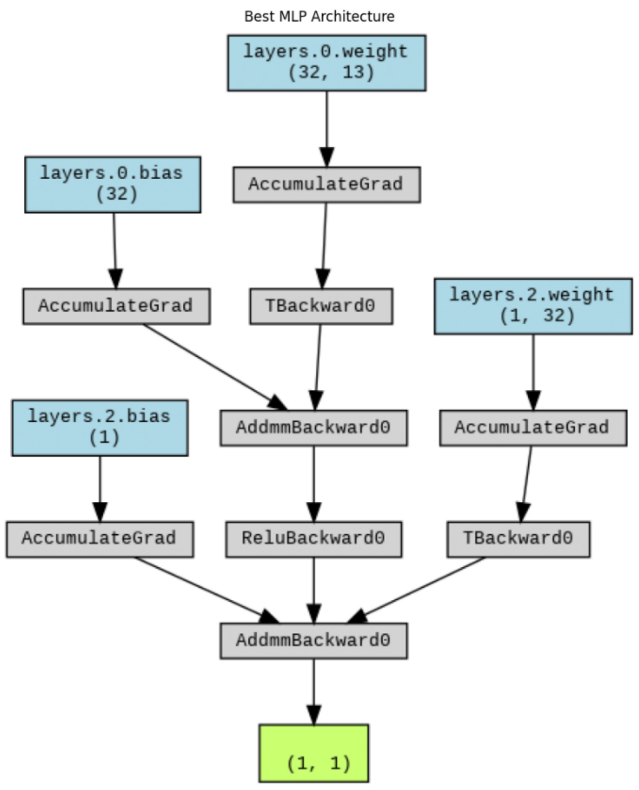


Figure 3.8: The Best Architecure

3.3 Class Prediction - CNN

Dataset and Classification of Trusses

The dataset used to predict the classes of truss structures consists of annotated images representing various truss configurations. Each image is labeled to indicate whether the structure is stable or unstable. Stable structures are those that effectively distribute loads and maintain their integrity under various loading conditions. In contrast, unstable structures are prone to deformation or collapse under certain loads.

Classes in the Dataset:

Stable Class: These images represent trusses with geometric configurations and material properties that ensure even load distribution, avoiding failure points -Class 0.
unstable Class: These images show trusses with structural weaknesses, excessive deformations, or potential failure points under applied loads -Class 1.

We will preprocess the images first, then use Convolutional Neural Networks (CNN) to predict the class of each truss. CNNs can detect complex patterns and subtle variations, making them ideal for distinguishing between stable and unstable structures.

3.3.1 Image Preprocessing for Truss Structure Classification

In our image preprocessing workflow, we aim to enhance the quality of input images before feeding them into the Convolutional Neural Network (CNN) model. This involves a series of key steps:

1. **Conversion to Grayscale:** Initially, we convert the images to grayscale to simplify processing and reduce computational load.
2. **Gaussian Blur:** We then apply Gaussian blur to smoothen the images and improve contour detection accuracy.
3. **Thresholding:** Following this, thresholding techniques are employed to create binary images, emphasizing truss structures while minimizing background noise.
4. **Morphological Operations:** Morphological operations like erosion and dilation are used to further refine the binary images, enhancing truss contour clarity.
5. **Contour Identification:** Next, we identify and isolate the largest contour, representing the truss structure, to focus exclusively on relevant features.
6. **Image Cropping:** Finally, the images are cropped around the isolated contour, eliminating extraneous background and ensuring that only truss structures are included in the input data for the CNN model.

These meticulous preprocessing steps aim to provide the CNN with clear, informative images, ultimately leading to improved accuracy and efficiency in classifying truss structures.

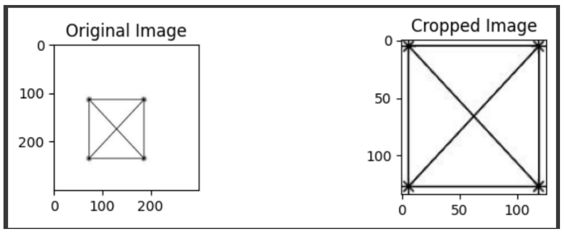


Figure 3.9: Cropping Image

3.3.2 Image Annotation

In this process, we annotate the dataset by classifying truss structures as either stable or unstable. We begin by dropping irrelevant columns from the dataset. Then, we calculate the mean of each remaining column to establish a threshold for classification. The thresholding criterion can be formulated mathematically as follows:

$$\text{Threshold for column } i = |\text{mean}(X_i)|$$

where X_i represents the values in column i.

A classification function is applied to each row of the dataset, comparing the absolute values of the features to the corresponding column means. If a feature value falls below the column mean, it is classified as stable; otherwise, it is classified as unstable. The classification criterion can be represented as:

$$\text{Class}(row) = \begin{cases} 0 & \text{if } |\text{feature}(row)| \leq \text{Threshold for corresponding column} \\ 1 & \text{otherwise} \end{cases}$$

These classifications are added as a new column in the dataset. Finally, the annotated dataset is saved to a specified file path.

Pseudo Code

Algorithm 1 Data Annotation with Thresholding

```
Require: Data: input dataset, Output_file_path: path to save annotated dataset
 1: Drop irrelevant columns from Data
 2: Calculate the mean of each remaining column
 3: for each row in Data do
 4:     for each feature in the row do
 5:         if absolute value of feature $\leq$ Threshold for corresponding column then
 6:             Classify feature as stable
 7:         else
 8:             Classify feature as unstable
 9:         end if
10:     end for
11: end for
12: Add classification column to Data
13: Save Data to Output_file_path
```

3.3.1 Model Architecture

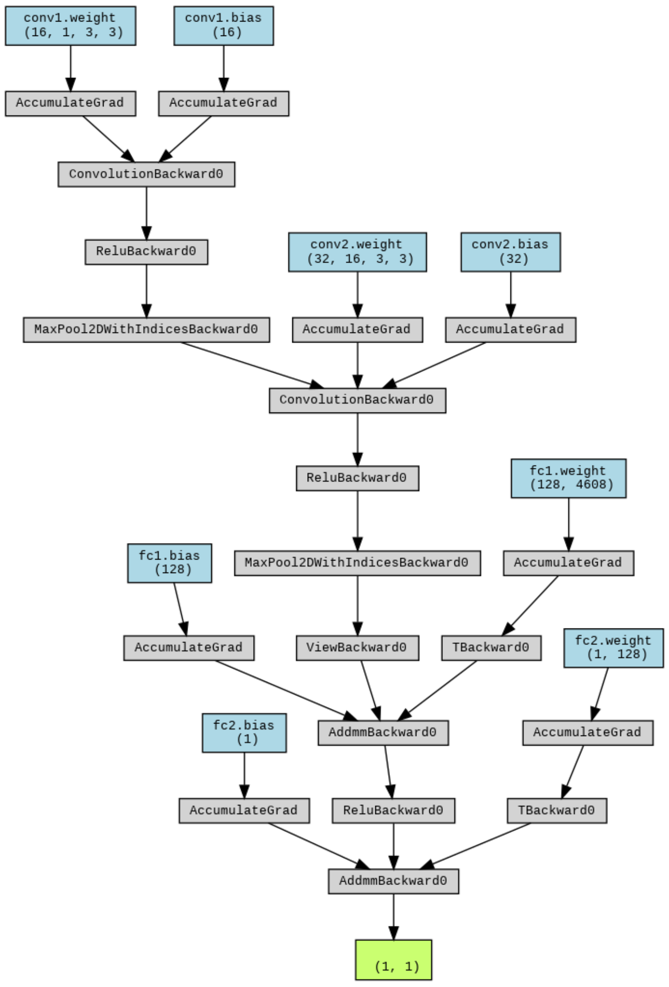


Figure 3.10: Model Architecture

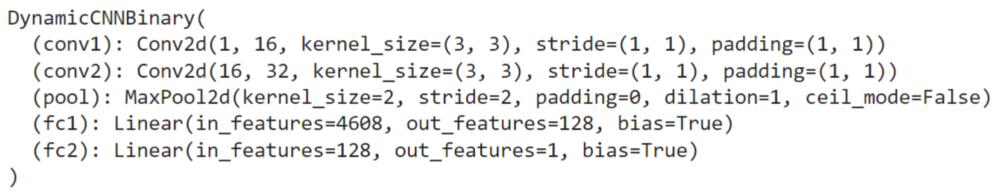
The architecture of our Convolutional Neural Network (CNN) model, named `DynamicCNNBinary`, is specifically designed for the binary classification of truss structures into stable and unstable categories. The model comprises several layers as follows:

The first layer, `conv1`, is a convolutional layer that applies 16 filters to the input image, each of size 3x3, with a stride of 1 and padding of 1. This configuration ensures that the spatial dimensions of the output are preserved. The second layer, `conv2`, is another convolutional layer with 32 filters of size 3x3, also with a stride of 1 and padding of 1, increasing the depth of the feature maps to capture more complex patterns.

Following the convolutional layers is a pooling layer, `pool`, which applies a max-pooling operation with a 2x2 kernel and a stride of 2. This layer reduces the spatial dimensions of the feature maps by a factor of 2, effectively downsampling the data and reducing the computational load for subsequent layers.

The output from the pooling layer is then flattened and passed through the first fully connected layer, `fc1`, which consists of 128 neurons. This layer learns to combine the features extracted by the convolutional layers. Finally, the output is passed through the second fully connected layer, `fc2`, which has a single output neuron that produces a binary output indicating the class of the input truss structure (stable or unstable).

This layered architecture allows the model to effectively learn and differentiate between stable and unstable truss structures by extracting relevant features from the input images, reducing dimensionality and computational complexity, and performing the final classification with high accuracy.

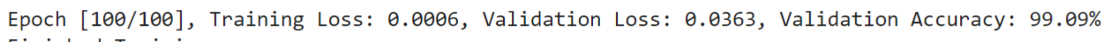


```
DynamicCNNBinary(
  (conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=4608, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=1, bias=True)
)
```

Figure 3.11: Model Architecture

3.3.2 Training and results

In the training process, a Convolutional Neural Network (CNN) was utilized to predict the class membership of truss structures. The model was trained for 100 epochs, achieving notable results in terms of classification accuracy and loss reduction. After 100 epochs, the training loss was reduced to 0.0006, while the validation loss decreased to 0.0363. Impressively, the validation accuracy reached 99.09%, indicating that the model effectively learned the patterns in the data and generalized well to unseen examples. These results demonstrate that the CNN successfully captured the underlying relationships between the input features and the target class labels, showcasing its efficacy in handling the complex task of classifying truss structures. The high validation accuracy further confirms the model's robustness and reliability in real-world applications.

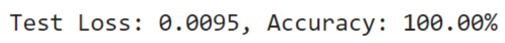


```
Epoch [100/100], Training Loss: 0.0006, Validation Loss: 0.0363, Validation Accuracy: 99.09%
```

Figure 3.12: Results

3.3.3 Test Result

The trained Convolutional Neural Network (CNN) was subsequently evaluated on a test set to assess its generalization performance. The results were outstanding, with the model achieving a test loss of 0.0095 and a perfect accuracy of 100.00%. This indicates that the CNN not only performed exceptionally well on the training and validation datasets but also flawlessly generalized to new, unseen data. The low test loss and perfect accuracy underscore the model's robustness and its ability to accurately classify truss structures in practical scenarios.



```
Test Loss: 0.0095, Accuracy: 100.00%
```

Figure 3.13: Test result

3.4 Problem of Data Imbalance

Data imbalance is a common issue in machine learning and deep learning tasks, particularly in classification problems. It occurs when the number of instances in one class significantly outnumbers the instances in other classes. This imbalance can lead to several challenges, such as biased predictions towards the majority class and poor generalization for the minority class.

In our dataset, there's an imbalance between the classes, with the majority of instances belonging to the stable class (Class 0) and fewer instances representing the unstable class (Class 1). This class imbalance poses a challenge as it may lead to the model being biased towards predicting the majority class, neglecting the minority class instances that are equally important.

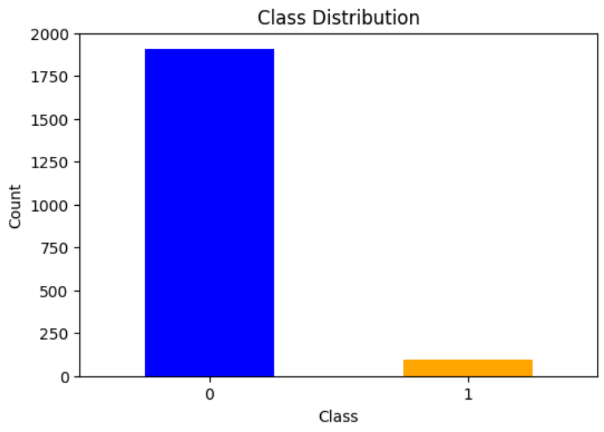


Figure 3.14: Imbalanced Distribution

3.5 Data Augmentation

Data augmentation is a critical technique in machine learning and deep learning aimed at increasing the diversity and size of the training dataset. Its primary objective is to enhance the generalization ability of machine learning models by exposing them to a wider range of variations in the input data. This helps improve the robustness of the models and reduces overfitting, particularly when dealing with limited training data.

To address class imbalance and increase the number of samples in the minority class, several techniques are employed. These techniques aim to generate new synthetic samples while preserving the semantic information of the original data:

- **Random Rotation:** Rotate the images by a certain degree to introduce variability.
- **Horizontal and Vertical Flips:** Flip the images horizontally or vertically to create new perspectives.
- **Random Resized Crop:** Crop and resize the images to different sizes to simulate variations in scale.
- **Color Jittering:** Randomly adjust the brightness, contrast, saturation, and hue of the images to change their appearance.

3.5.1 Types of Data Augmentation

Rotation

Rotation involves rotating the image by a certain angle. The transformation matrix for rotation by an angle θ is given by:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Translation

Translation involves shifting the image by a certain distance in the x and y directions. The transformation matrix for translation by distances t_x and t_y is given by:

$$T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

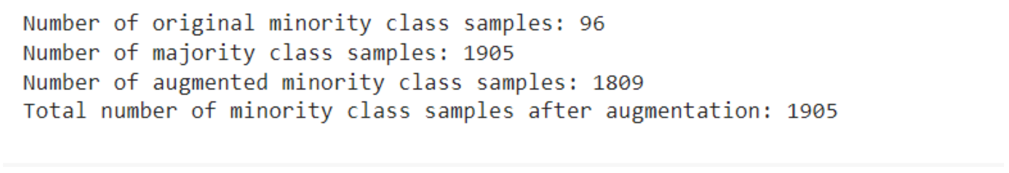
Scaling

Scaling involves resizing the image by a certain factor. The transformation matrix for scaling by factors s_x and s_y is given by:

$$S(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

3.6 Results On balanced data

Here the dataset after augmentation :



```
Number of original minority class samples: 96
Number of majority class samples: 1905
Number of augmented minority class samples: 1809
Total number of minority class samples after augmentation: 1905
```

Figure 3.15: Augmented Dataset

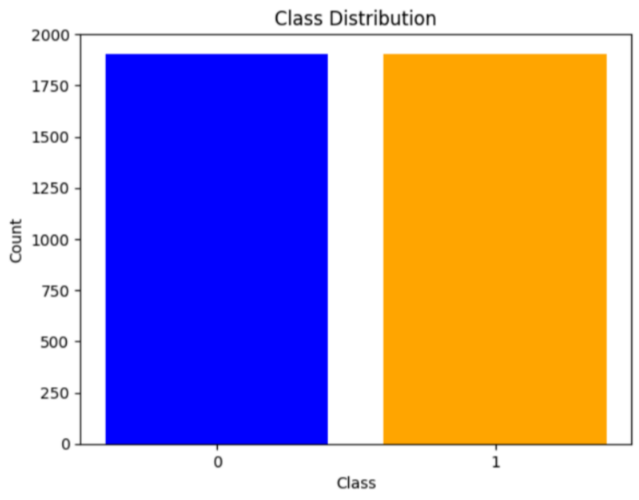
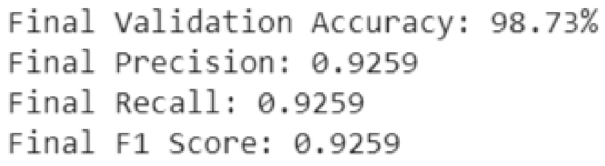


Figure 3.16: Data Distributioin



```
Final Validation Accuracy: 98.73%
Final Precision: 0.9259
Final Recall: 0.9259
Final F1 Score: 0.9259
```

Figure 3.17: Train With Final Dataset

3.6.1 Training Results

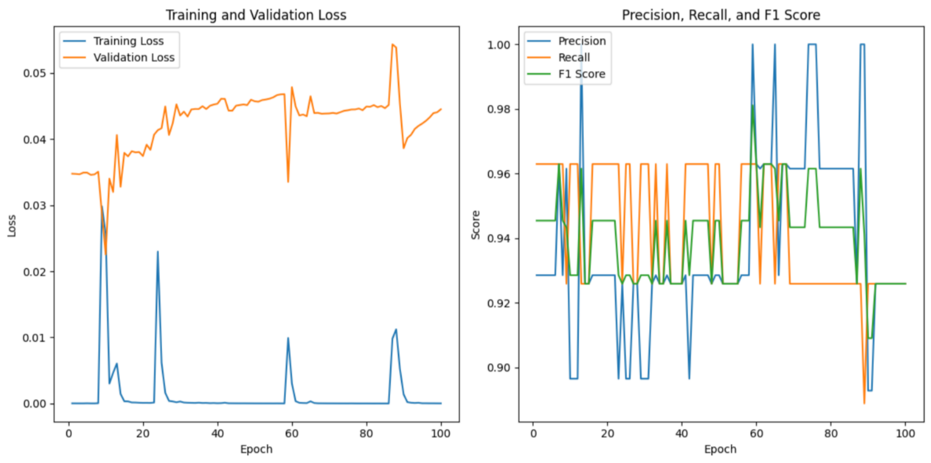
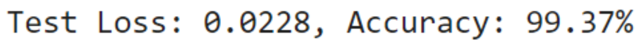


Figure 3.18: Training Results

3.6.2 Test Results



```
Test Loss: 0.0228, Accuracy: 99.37%
```

Figure 3.19: Test With balanced Dataset

When trained on the balanced dataset, the model exhibits slightly different characteristics.We obtained a training loss of 0.0003 and the validation loss decreased to 0.0211, and the validation accuracy remained consistently high at 99.04%. This suggests that the model's performance generalizes well to new data even with a balanced distribution of classes. Notably, the test loss reduced to 0.0228, and the test accuracy increased to an impressive 99.37%, surpassing the performance on the imbalanced dataset. This improvement could be attributed to the model's ability to learn more robust features across all classes in the balanced dataset, mitigating the risk of overfitting to any particular class. Overall, these results underscore the importance of dataset balance in achieving optimal performance and generalization in CNN-based image classification tasks.

4 Conclusion

In this project, we aimed to leverage Convolutional Neural Networks (CNNs) for predicting structural properties and classifying truss configurations. By harnessing the power of deep learning, we sought to provide accurate predictions for weight and energy values of truss structures while also classifying them as stable or unstable based on their configurations.

Our approach involved meticulous preprocessing of input data, ensuring that the CNN models received clear and informative images to make accurate predictions. Additionally, we utilized Genetic Algorithms (GA) to optimize the neural network architecture, enhancing the models' predictive capabilities.

Throughout the implementation, we encountered challenges such as data imbalance, which we addressed through data augmentation techniques to improve model generalization and mitigate bias towards the majority class.

The results obtained were highly promising, with the CNN models demonstrating exceptional performance in both prediction tasks and classification. Notably, the models achieved low training, validation, and test losses, as well as high accuracies, indicating their robustness and reliability in real-world applications.

Overall, this project showcases the potential of deep learning techniques, particularly CNNs, in engineering tasks, offering accurate predictions and classifications for truss structures. These findings have significant implications for structural analysis and design, paving the way for more efficient and reliable engineering solutions.

Bibliography

[1] Prince, S.J.D. (2024). *Understanding Deep Learning*. Retrieved from `http://udlbook.com`

[2] Using Genetic Algorithms to Train Neural Networks. Retrieved from `https://towardsdatascience.com/using-genetic-algorithms-to-train-neural-networks-b5ffe0d51321`