Sudoku Solver Project Report

Menashe Ashkenazi and Asaf Shemesh

Computer Science, Ariel university

20th January 2025

Abstract

This report presents our work on solving Sudoku puzzles using deep learning techniques, leveraging a custom dataset of 2500 Sudoku images. The project investigates several deep learning models, including baseline, logistic regression, simple neural networks, and advanced convolutional neural networks (CNNs), to address the problem of digit recognition and puzzle completion. Each model is evaluated based on its performance, with results presented through both quantitative metrics (accuracy, precision, and recall) and visual demonstration of solved Sudoku grids. Key challenges encountered during the project, such as image preprocessing and model optimization, are discussed, along with the steps taken to improve model accuracy. The complete project code is available on GitHub at `https://github.com/asafShemesh/sudoku_deepLearning`.

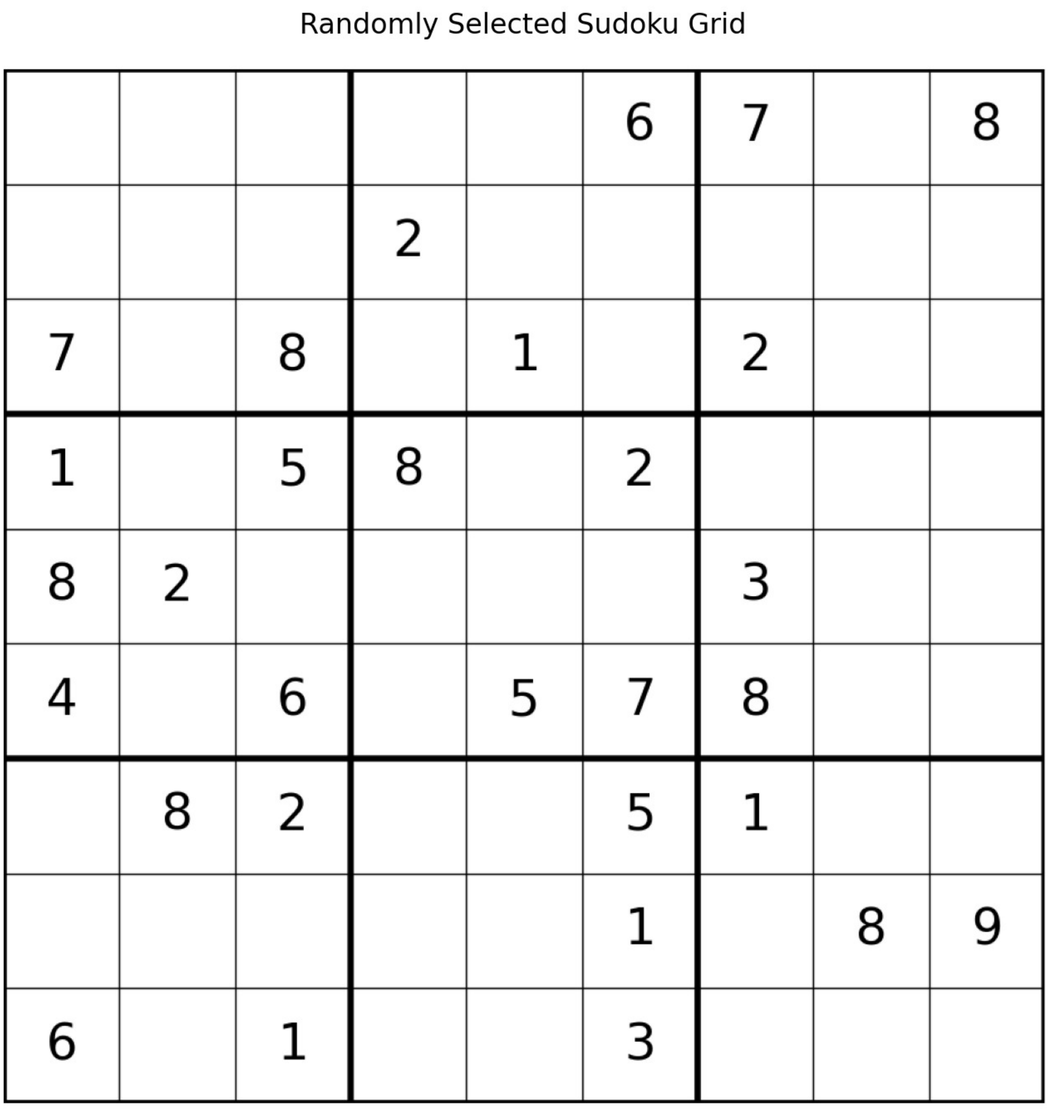
1 Introduction

1.1 Background and Motivation

The goal of this project is to solve Sudoku puzzles using deep learning techniques. Given a set of images containing partially filled Sudoku grids, the task is to recognize the digits in each cell and complete the puzzle while adhering to the rules of Sudoku. The project explores multiple deep learning models to solve the puzzle.

1.2 Dataset and Data Preparation

The dataset for this project consists of 2500 images of Sudoku grids, which were generated using a Python program we developed.This program is located in file named `'Generate_dataset.py'`. These images were then preprocessed to extract the grid and individual digits for model training.


Randomly Selected Sudoku Grid

					6	7		8
			2					
7		8		1		2		
1		5	8		2			
8	2					3		
4		6		5	7	8		
	8	2			5	1		
					1		8	9
6		1			3			

Figure 1: Example of data

First, the border of the grid is identified by finding the largest contour in each image. This step is essential as it helps isolate the grid from the rest of the image, ensuring that only the relevant portion of the image is processed. The process begins by converting the image to grayscale and applying a binary threshold to distinguish the grid from the background. Using OpenCV's 'findContours', the contours of the image are detected, and the largest contour is selected, which is assumed to represent the grid. The bounding rectangle of this contour is then calculated using ‘cv2.boundingRect‘, which provides the coordinates of the grid’s edges (x, y) and its dimensions (width, height). These coordinates are used to crop the image, isolating the grid for further processing. Once the grid is cropped, it is resized to 450x450 pixels to standardize the input size. The next step involves splitting the grid into individual 9x9 cells. Each cell is resized to 28x28 pixels, which is the standard input size for the digit recognition models.

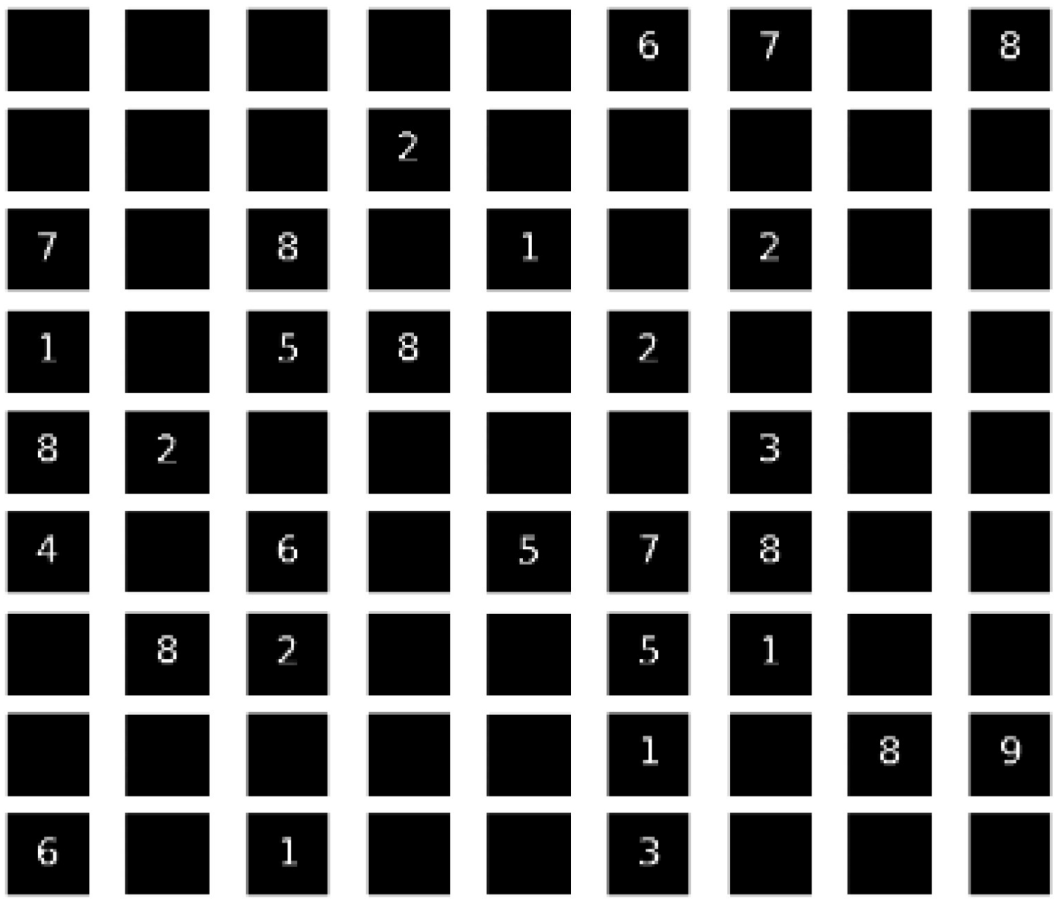


Figure 2: Sudoku after preprocessing and cell splitting.

The cells are normalized and converted into tensors to ensure consistency in the data fed into the neural network models. The processed dataset is then split into 80% training and 20% testing sets to evaluate the performance of the models.

1.3 Models and Objectives

This project explores various models for solving Sudoku puzzles, starting with simple methods like baseline and logistic regression and progressing to more complex architectures, such as basic neural networks and convolutional neural networks (CNNs). The baseline model allocates random values to the masked regions, serving as the simplest reference point. More advanced models, such as logistic regression and neural networks, aim to learn more complex patterns within the grid. Finally, the CNN model extracts hierarchical features to further improve digit recognition.

The goal is to determine the most effective approach for solving Sudoku puzzles by comparing how each model handles digit identification and puzzle-solving while adhering to Sudoku rules.

2 Models

2.1 Baseline Model

The baseline model randomly assigns digits to the Sudoku grid.

2.1.1 Results

Accuracy: 9.88%
Precision: 9.88%
Recall: 9.88%

2.2 Logistic Regression Model

The logistic regression model flattens the 28x28 pixel cells into a vector and uses the logistic regression algorithm to predict digits (0-9) for each cell. This model treats each cell as an independent feature and performs multi-class classification for each digit.

2.2.1 Training

The model is trained using the logistic regression algorithm with the 'lbfgs' solver and a maximum of 1000 iterations. The dataset is split into training and test sets (80% training and 20% testing). The model uses a cross-entropy loss function to minimize the prediction errors. A graph is plotted to visualize the training process:

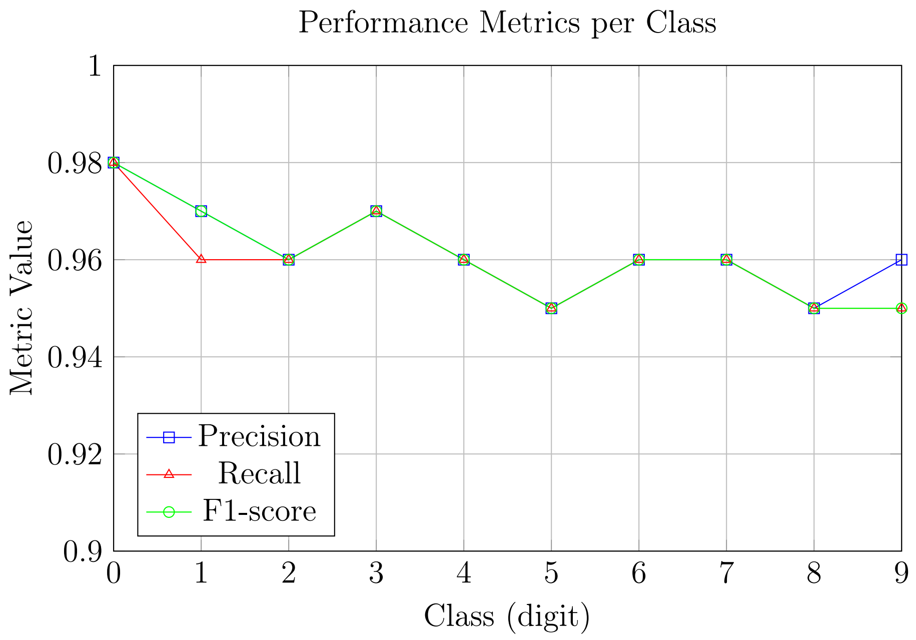


Figure 3: Performance metrics per class for the classifier.

2.2.2 Results

Accuracy: 97%
Precision: 97%
Recall: 96%

2.3 Basic Neural Network

The model includes an input layer that converts a 28x28 image into a 784-pixel vector, followed by two fully connected layers with 128 and 64 neurons. The final layer outputs predictions for digits 0-9 with 10 neurons.

2.3.1 Optimizing with Adam and Cross-Entropy Loss

In this model, the Adam optimizer is used to minimize the loss function during training. Adam adapts the learning rate based on the first and second moments of the gradients, making it efficient for large datasets and preventing overfitting. It helps the model converge faster than traditional gradient descent. The cross-entropy loss function is employed because it is ideal for multi-class classification tasks like digit recognition. It measures the difference between the predicted probabilities and the actual labels, guiding the model to improve its predictions with each iteration. The combination of Adam and cross-entropy allows the model to learn effectively and accurately from the data.

2.4 Training

The model is trained using the Adam optimizer and cross-entropy loss to minimize prediction errors.

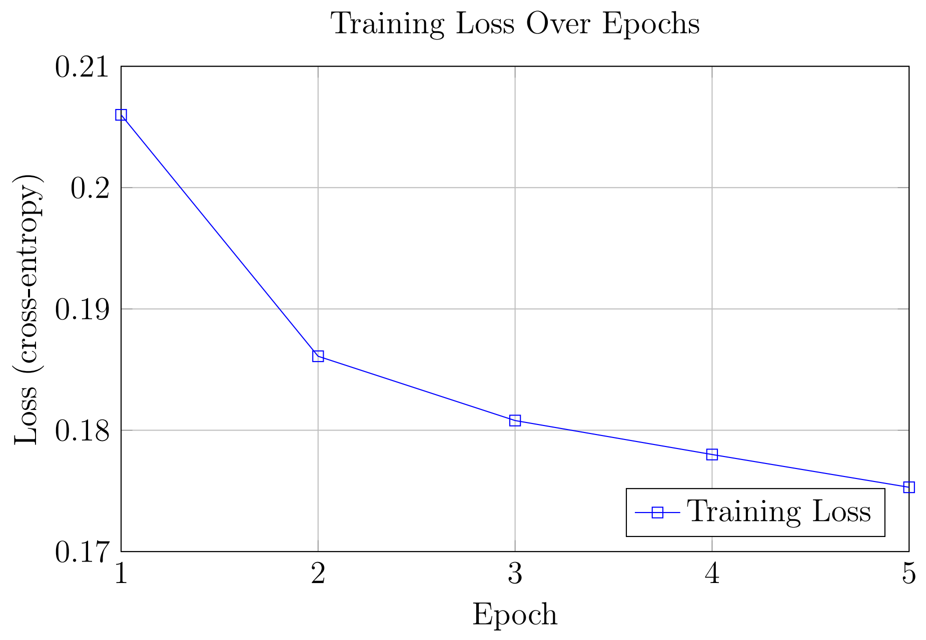


Figure 4: Training loss curve for the neural network.

2.4.1 Results

Accuracy: 97%
Precision: 97%
Recall: 97.49%

2.5 Advanced CNN Model

The CNN model enhances performance in digit recognition by using multiple convolutional layers for feature extraction. Max-pooling and batch normalization help stabilize the learning process, while dropout regularizes the fully connected layers to prevent overfitting.

2.5.1 CNN Architecture

The model processes the input through the following steps:

1. **Convolutional Layers:** The model begins with four convolutional layers: ‘conv1‘, ‘conv2‘, ‘conv3‘, and ‘conv4‘. These layers extract key features from the input images by applying filters to detect patterns like edges and textures.

2. **Batch Normalization:** Each convolutional layer is followed by batch normalization layers (‘bn1‘, ‘bn2‘, ‘bn3‘, ‘bn4‘). These layers normalize the activations, stabilizing the learning process and improving model performance by reducing internal covariate shift.

3. **Max-Pooling:** Max-pooling is applied after each convolutional block. It reduces the spatial dimensions of the feature maps, helping the model focus on important features while reducing computation.

4. **Fully Connected Layers:** The output from the convolutional and pooling layers is flattened and passed through fully connected layers (‘fc1‘, ‘fc2‘, ‘fc3‘).

    - ‘fc1‘ has 512 neurons.
    - ‘fc2‘ has 128 neurons.
    - ‘fc3‘ outputs 10 values, corresponding to the digits 0-9.

5. **Dropout:** Dropout is applied after the first two fully connected layers to prevent overfitting. By randomly deactivating a portion of neurons during training, it helps the model generalize better on unseen data.

2.5.2 Training

The model is trained using the AdamW optimizer with cross-entropy loss to minimize prediction errors. AdamW is a variant of the Adam optimizer that decouples weight decay from the gradient updates, leading to better generalization performance by preventing overfitting.

2.5.3 Results

Accuracy: 97%
Precision: 97%
Recall: 97.49%

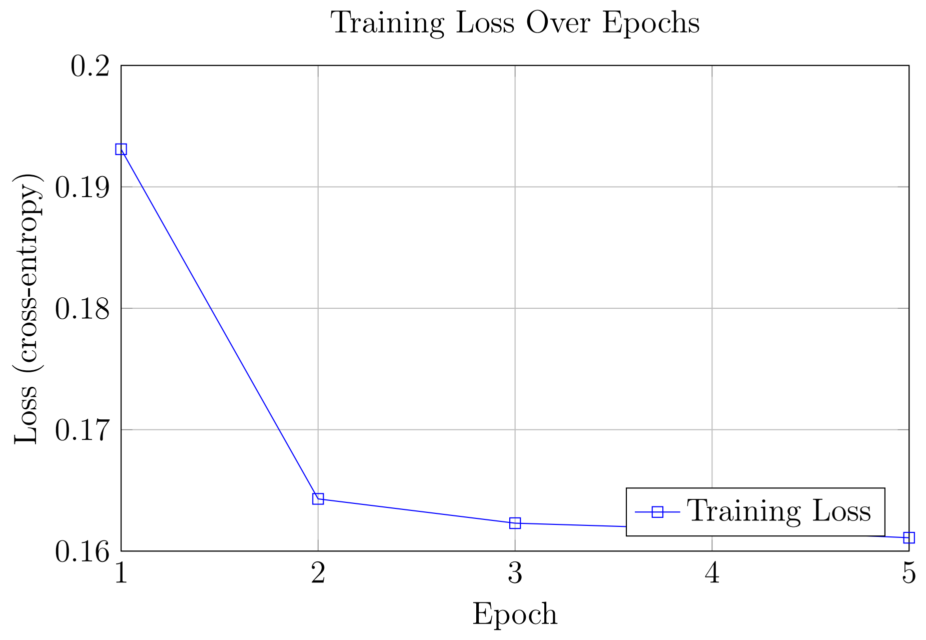


Figure 5: Training loss curve for the neural network.

3 Conclusion

This project demonstrates the effectiveness of various models for Sudoku digit recognition and reconstruction. The results show that the fully connected model, despite its simplicity, performs reasonably well in reconstructing masked regions. However, more complex models, such as the CNN, significantly outperform the basic models, demonstrating the advantages of deeper architectures for image recognition tasks. The graph below shows the comparison between the four models used in this project.

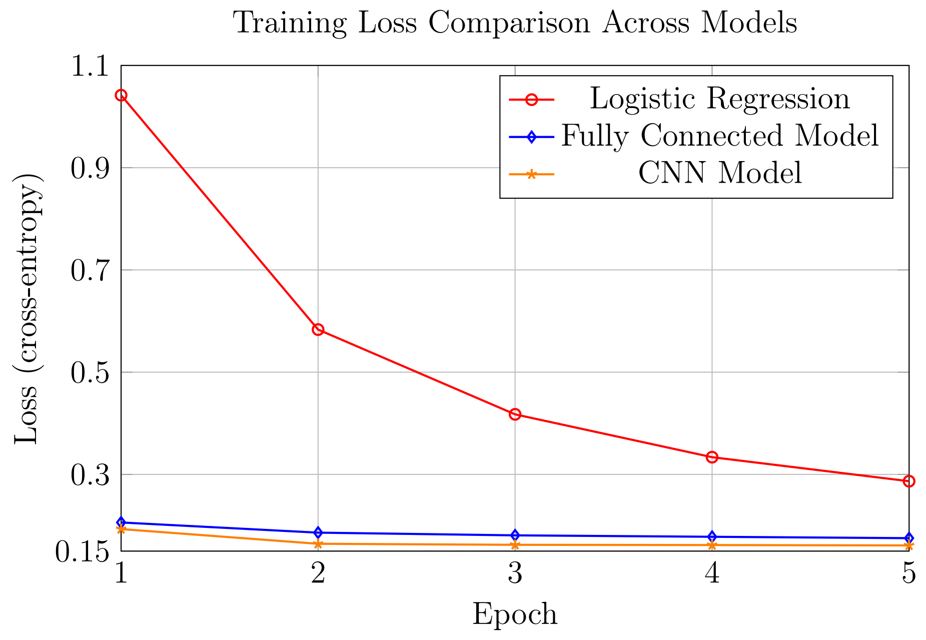
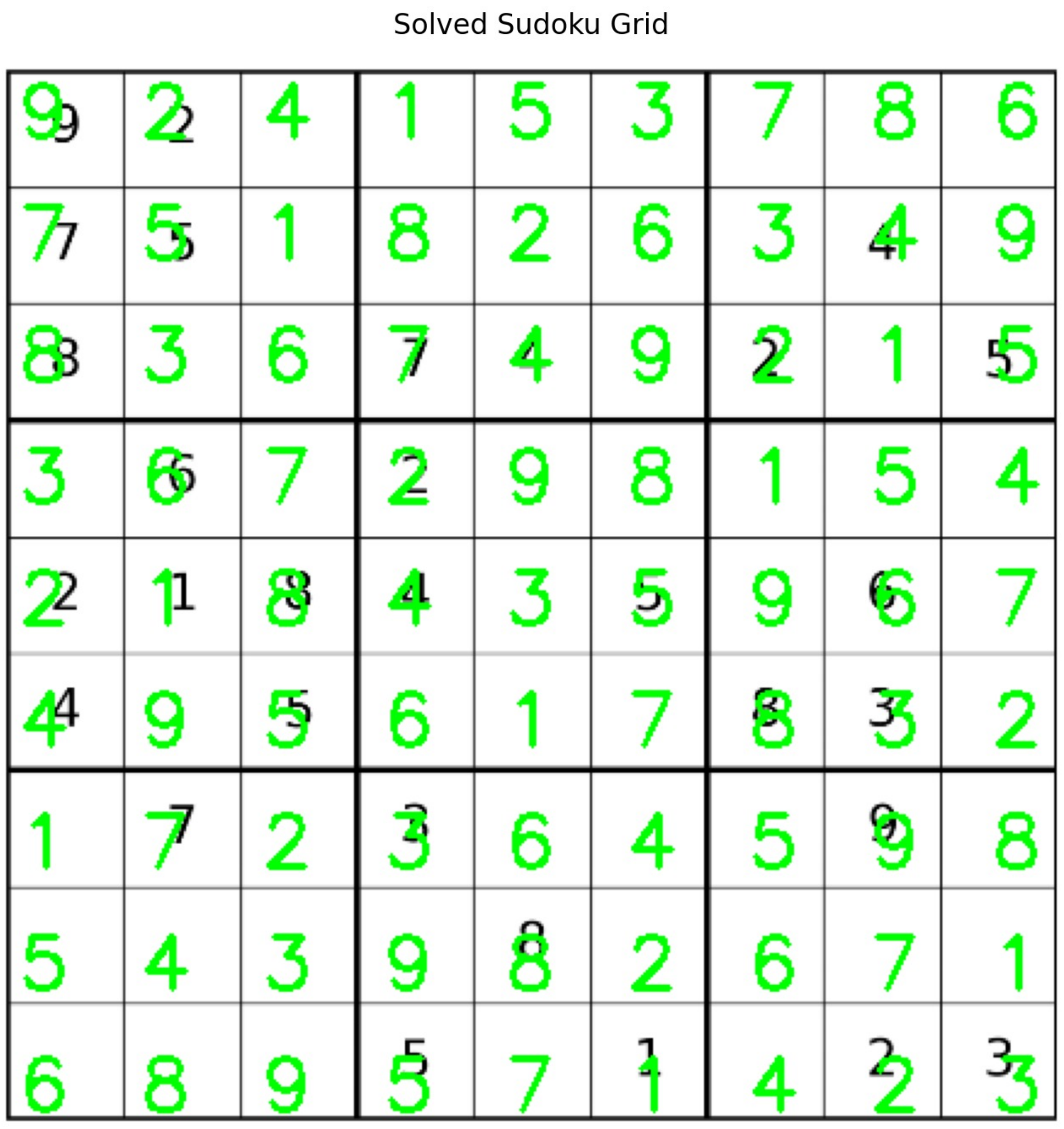


Figure 6: Comparison of training loss across different models.

3.1 Model Performance

The image shows a solved Sudoku grid, where the black numbers represent the initial given digits, and the green numbers indicate the predictions made by the model.


Solved Sudoku Grid

9	2	4	1	5	3	7	8	6
7	5	1	8	2	6	3	4	9
8	3	6	7	4	9	2	1	5
3	6	7	2	9	8	1	5	4
2	1	8	4	3	5	9	6	7
4	9	5	6	1	7	8	3	2
1	7	2	3	6	4	5	9	8
5	4	3	9	8	2	6	7	1
6	8	9	5	7	1	4	2	3

Figure 7: Full Sudoku prediction