Neural Network Implementation for Predicting Medical Appointment No-Shows

Mohit Gunani
Electronics Engineering

June 1, 2025

Contents

1 Introduction

This report presents a comparative study of two neural network implementations for predicting patient no-shows in medical appointments using the same dataset:

1. A model implemented from scratch using Python and NumPy.
2. A model built using Tensorflow-Keras.

We evaluate both models in terms of convergence, accuracy, memory usage, and overall effectiveness.

2 Dataset and Preprocessing

The dataset contains information on over 100,000 medical appointments, including features like age, gender, neighbourhood, scheduled day, appointment day, and whether a patient showed up.

- Removed irrelevant columns (e.g., patient ID,appointment id).
- Categorical features were one-hot encoded.
- Numerical features were normalized.
- Target variable: `No-show` (Yes/No).

3 Model Architectures

3.1 From-Scratch Implementation

Built using only Python and NumPy, the architecture was:

- Input layer: 89 features
- Hidden layer: 10 neurons with ReLU activation
- Output layer: 1 neuron with sigmoid activation

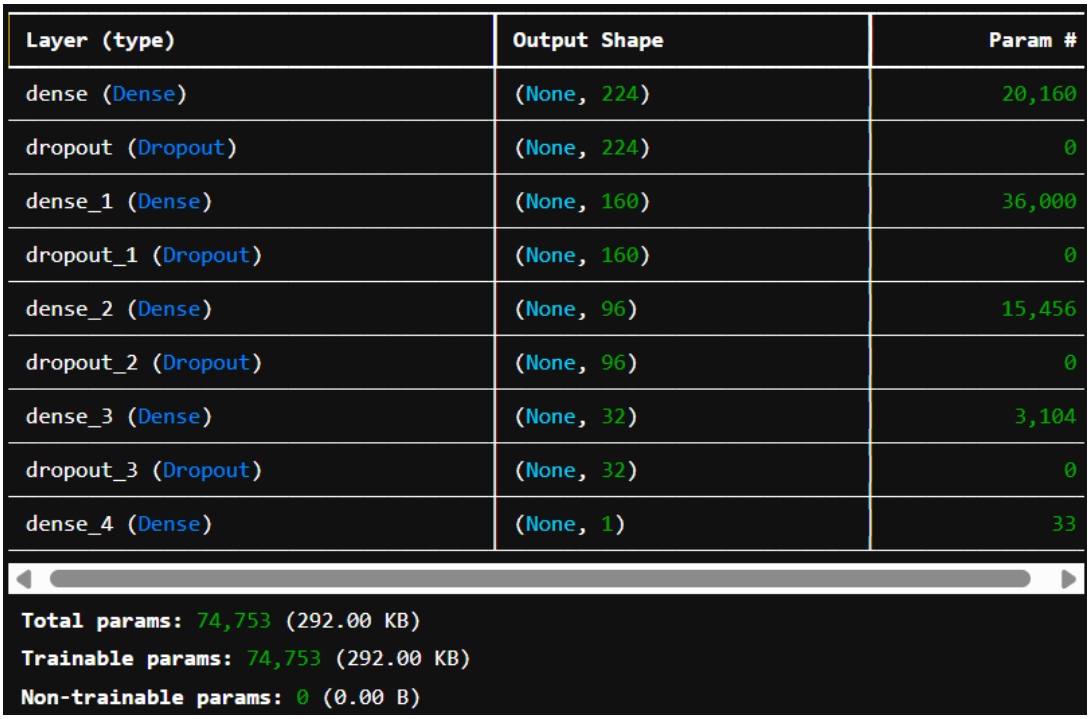
Gradient descent and binary cross-entropy loss were implemented manually. No external ML libraries were used.

3.2 Keras Implementation

The Keras model used a deeper architecture with dropout to improve generalization:

- Dense (224) + Dropout
- Dense (160) + Dropout
- Dense (96) + Dropout
- Dense (32) + Dropout
- Dense (1) with sigmoid activation

Total trainable parameters: 74,753



Layer (type)	Output Shape	Param #
dense (Dense)	(None, 224)	20,160
dropout (Dropout)	(None, 224)	0
dense_1 (Dense)	(None, 160)	36,000
dropout_1 (Dropout)	(None, 160)	0
dense_2 (Dense)	(None, 96)	15,456
dropout_2 (Dropout)	(None, 96)	0
dense_3 (Dense)	(None, 32)	3,104
dropout_3 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 1)	33

Figure 1: Keras Model Summary

4 Training Configuration

- Optimizer: Adam
- Loss Function: weighted Binary Crossentropy
- Epochs: 72 (with early stopping)
- Batch Size: 32
- Validation Data : Xtest, ytest

5 Evaluation and Analysis

5.1 1. Convergence Time

- **From-Scratch:** Converged in 1000 epochs.
- **Keras:** Converged in 24 epochs with early stopping.
- *Reason:* Keras benefits from optimized tensor operations and auto-differentiation.

5.2 2. Performance Metrics

Metric	From Scratch	Keras
Accuracy	61.27%	54.1%
F1 Score	0.4198	0.4325
PR-AUC	0.5778	0.5778

5.3 3. Memory Usage

- From-Scratch: ˜327.31 MB peak RAM
- Keras: ˜249.51MB due to backend + graph overhead

5.4 4. Confusion Matrix and Inference

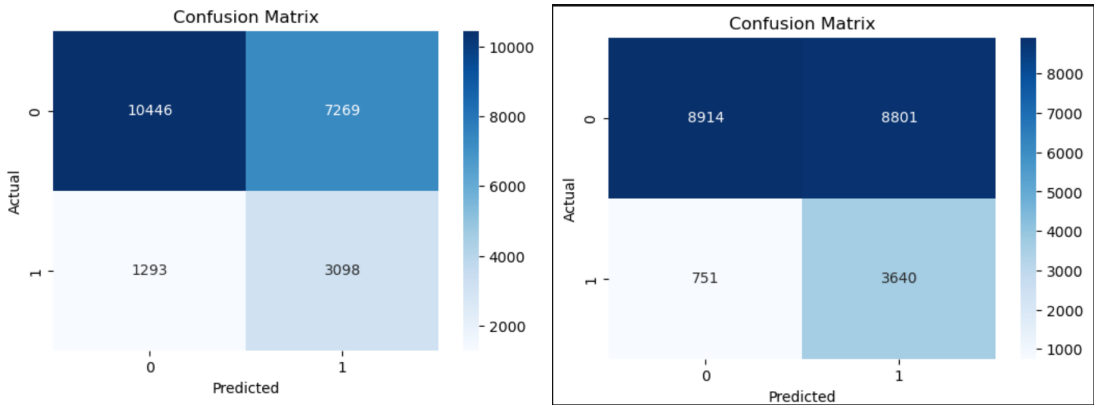


Figure 2: Confusion Matrices: From-Scratch (left) vs. Keras (right)

Keras model shows fewer false negatives, making it more reliable for identifying no-shows.

5.5 5. Analysis and Discussion

- **Convergence:** Keras trains faster due to vectorization and hardware acceleration.
- **Performance:** Improved due to deeper layers and dropout regularization.
- **Numerical Stability:** Keras handles initialization and loss gradients robustly.
- **Code Complexity:** The scratch model helps in understanding internals; Keras is faster to develop and scale.

6 Conclusion

The from-scratch implementation provides valuable learning, but the Keras model clearly outperforms in training speed, performance, and reliability. Keras proves to be a practical choice for real-world deployment, while the from-scratch model is a strong educational tool.