University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Building Neural Networks and CNN

Assignment-3

Jayant Som (jsom : 50625536)
Rithvik Illandula (rithviki : 50626084)

Department of Computer Science and Engineering
University at Buffalo

CSE 574: Introduction to Machine Learning

July 02, 2025

Part I : Building a basic NN

1.1 Dataset Description

The income dataset we used to predict whether an individual's income exceeds $50K per year.

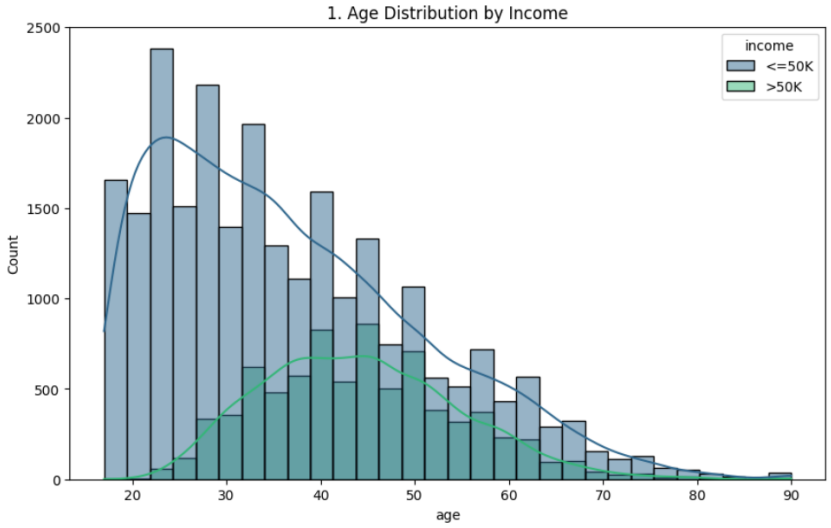
- **Rows**: We have noticed rows around 32,561
- **Columns**: There are 15 Columns
- **Target**: The income (given in binary: <=50K or >50K)
- **Feature Types**: 6 numerical and 8 categorical
- **Class Imbalance**: ~24% have income >50K in the given data

1.2. Main Statistics

- **Average age:** The average age of Users in the dataset is 38.6 years
- **Average hours worked per week:** People worked around 40.4 hrs
- **Gender distribution:** Here in the dataset, there were 67% males and 33% females
- **Most common education:** Most of them were HS-grad (32%)
- **Most frequent occupation:** In the given information, the Prof-speciality (12.7%)
- **Most prevalent marital status:** According to the stats of the dataset, Married-civ-spouse (46%)
- **Most persistent work-class:** People work in the Private Sector with a great percentage of (69.7%)
- **Most common race:** Most people are White with (85.4%)

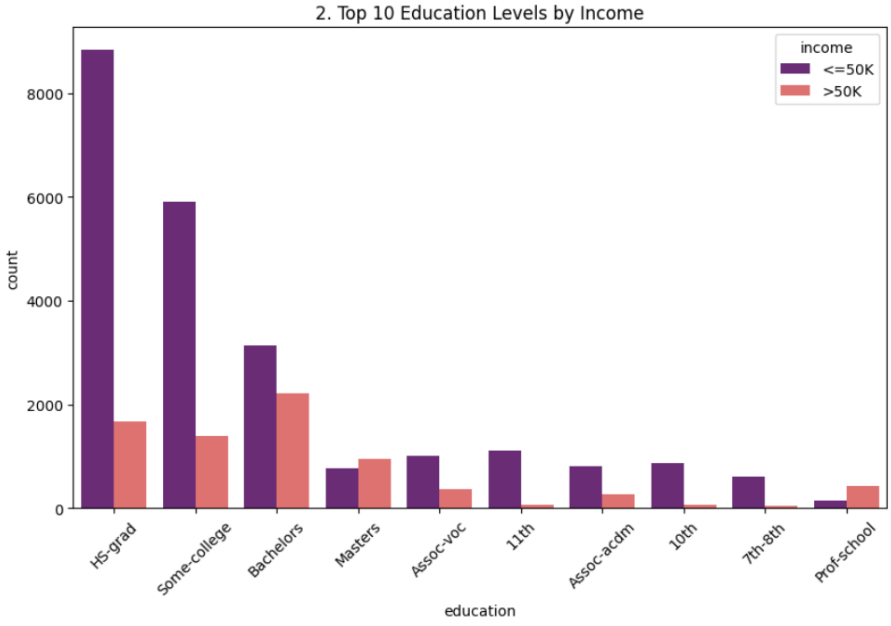
2. Data Visualizations

a. Age Distribution by Income



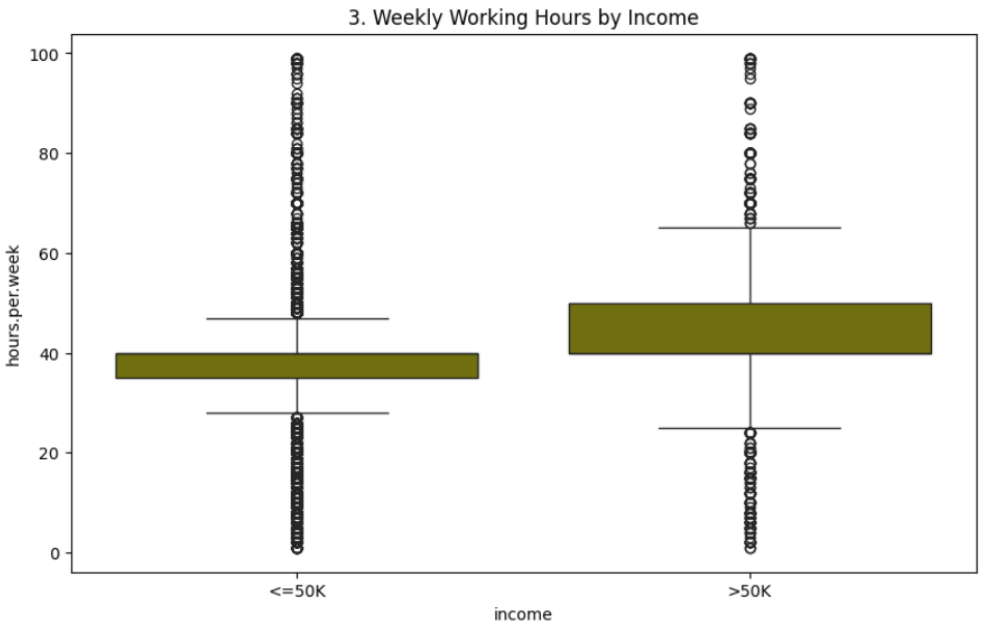
Here we have noticed that users have Peak earnings are exceeding $50,000 are seen around age 45. In contrast, the peak for incomes below $50,000 is observed at the age of 25.

b. Top 10 education levels by income



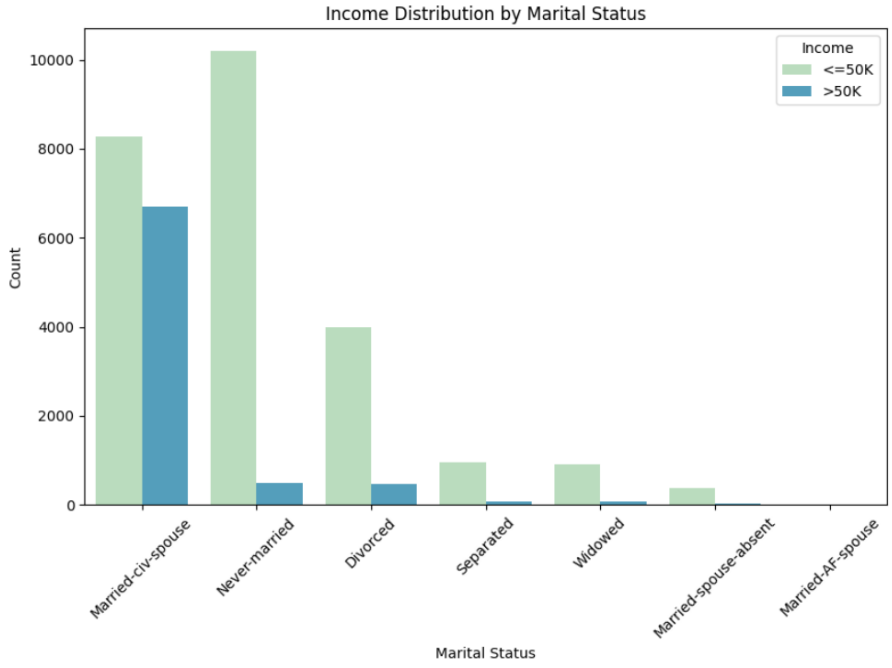
Here we can see that they have income with highest number of individuals earning over $50,000 possess a bachelor's degree, while the largest group earning under $50,000 consists of high school graduates.

c. Weekly Working Hours by Income



We can observe that on an average, people earning less than $50,000 work less hours per week as compared to the people earning more than $50,000.

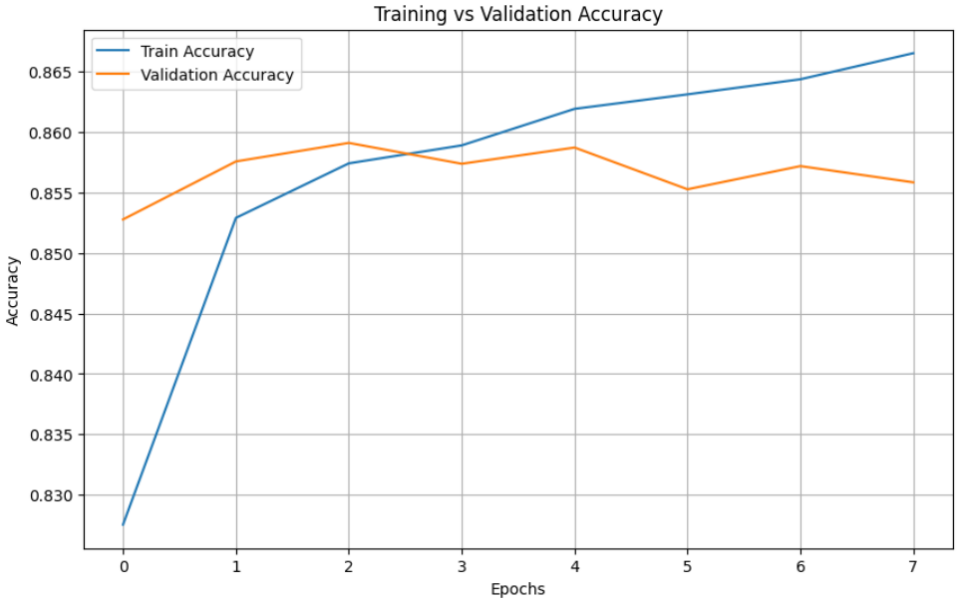
d. Income Distribution by Marital Status



Across all the marital statuses, the number of individuals earning over $50,000 is consistently lower than those earning less than $50,000.

4. Model Visualizations

a. Training vs Validation Accuracy



The training accuracy increased steadily across epochs and surpassed 86.5%, while the validation accuracy remained around 85.5%, indicating good generalization without overfitting.

b. Training vs Validation Loss

The Training loss decreased consistently, and validation loss stayed close, suggesting effective learning without underfitting or overfitting in the final outcome.

5. Preprocessing Summary

- Replaced all the entries with **NaN** and **dropped** incomplete rows.
- Used **OneHotEncoding** for all the categorical features.
- Scaled numeric features in the data using **StandardScaler.**
- We split data into 80% for training and 20% for test sets
- Final feature vector: We observed **108 dimensions**

Accuracy improvement tools:

1. Missing Value Handling was done to prevent the model from failing on missing data while preserving feature distribution.

2. OneHot Encoding was used to properly represent categorical relationships without false ordinality.

3. Feature Scaling was implemented to equalize feature influence for stable gradient descent.

6. Neural Network Architecture

Layer	Type	Neurons	Activation
Input	N/A	108	None
Hidden Layer 1	Dense	64	ReLU
Hidden Layer 2	Dense	32	ReLU
Hidden Layer 3	Dense	16	ReLU
Output	Dense	1	Sigmoid

- **Loss**: Binary Crossentropy
- **Optimizer**: Adam
- **EarlyStopping**: used to avoid overfitting

7. Results

- **Test Accuracy**: We have achieved accuracy of **86.06%** (Meets the minimum 82% requirement)

- The model shows us balanced performance across training and validation sets.

Part II: Building a CNN

2.1 Dataset Description

FASHION - MNIST is a benchmark dataset commonly we have used for image classification tasks, helping researchers to evaluate the models. It consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

Training data shape: (60000, 28, 28)
Test data shape: (10000, 28, 28)
Number of classes: 10

Type of Data:

- Image Data: Grayscale (1 channel) with a fixed size of 28×28 pixels.
- Labels: Integers (0–9), each representing a specific clothing category.

Variables (Features):

- Each image has 784 pixels (28×28), stored as a flattened array.
- Each pixel value ranges from 0 (black) to 255 (white).

Class distribution in training set:

T-shirt/top (0): 6000 samples
Trouser (1): 6000 samples
Pullover (2): 6000 samples
Dress (3): 6000 samples
Coat (4): 6000 samples
Sandal (5): 6000 samples
Shirt (6): 6000 samples
Sneaker (7): 6000 samples
Bag (8): 6000 samples
Ankle boot (9): 6000 samples

Key Characteristics:

- We observed “No color channels” (only grayscale).
- Small resolution (28×28), making it computationally efficient.
- Balanced dataset (each class has 6,000 training and 1,000 test samples).

2.2 Visualization

This visualization displays a 5×5 grid of 25 sample images from the Fashion-MNIST training set with their corresponding class labels.

2.3 Preprocessing Summary

1. Normalization: Scaled pixel values from [0, 255] to [0, 1] (float32).
2. Reshaping: Added channel dimension: (28, 28) → (28, 28, 1) (for CNN compatibility).
3. Label Encoding: Converted integer labels to one-hot vectors

2.4 CNN Architecture

Layer Type	Parameters / Details
Input	Shape = (28, 28, 1)
Conv2D	32 filters, 3×3 kernel, ReLU, padding="same"
MaxPooling2D	2×2 pool size
Conv2D	64 filters, 3×3 kernel, ReLU
MaxPooling2D	2×2 pool size
Dropout	Rate = 0.25
Flatten	Converts 3D to 1D
Dense	128 neurons, ReLU
Dropout	Rate = 0.5
Dense (Output)	10 neurons, Softmax

- **Loss**: Binary Crossentropy
- **Optimizer**: Adam
- **EarlyStopping**: used to avoid overfitting

Parameters:

Total parameters: 315,146
Trainable params: 315,146
Non-trainable params: 0

Input Neurons:

- The input shape is (28, 28, 1), meaning it accepts grayscale images of size 28×28 pixels.
- The number of input neurons is 28 × 28 × 1 = 784

Activation Functions:

- ReLU is used in the convolutional and dense hidden layers.
- Softmax is used in the output layer (for multi-class classification).

Hidden Layers:

1. Conv2D (32, (3,3)) + MaxPooling2D
2. Conv2D (64, (3,3)) + MaxPooling2D + Dropout
3. Dense (128) + Dropout

Total Hidden Layers: **3** (2 Conv + 1 Dense)

Kernel size, number of filters, strides, padding:

- **First Conv2D Layer:**

  Filters: 32
  Kernel size: 3×3
  Strides: Default (1,1) (not explicitly set)
  Padding: "same" (output size same as input)

- **Second Conv2D Layer:**

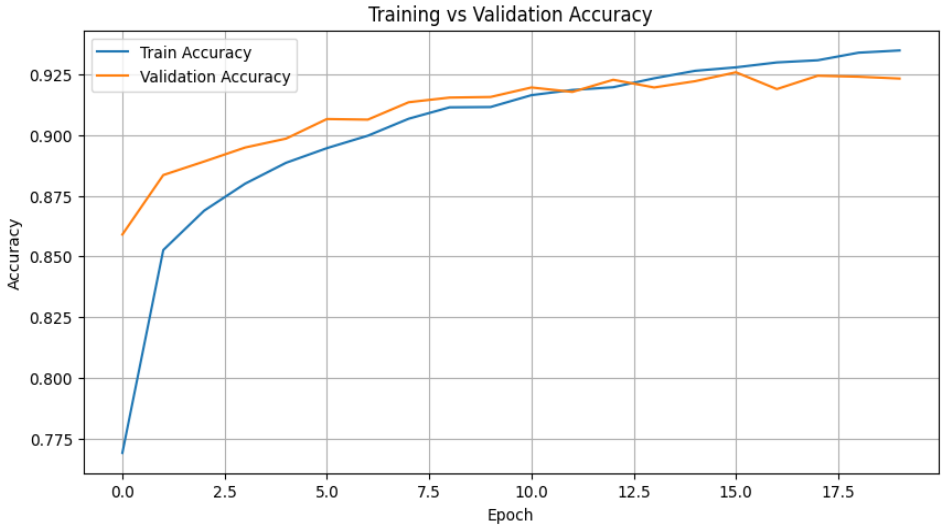
  Filters: 64
  Kernel size: 3×3
  Strides: Default (1,1)
  Padding: "valid" (default, reduces spatial dimensions)

- **MaxPooling2D Layers:**

  Pool size: 2×2
  Strides: Default (same as pool size, 2)

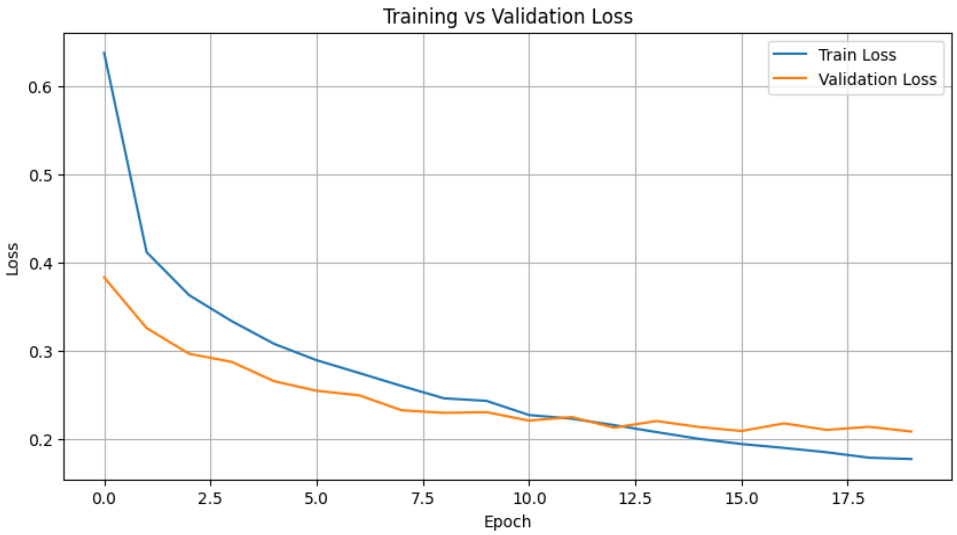
2.5 Plots

2.5.1 Training Vs Validation Accuracy



The training accuracy increased across epochs and steadily surpassed 92.5%, while the validation accuracy steadily increased and remained around 92.5%, indicating good generalization without overfitting.

2.5.2 Training Vs Validation Loss



Training loss decreased consistently, and validation loss stayed close, suggesting effective learning without underfitting or overfitting.

2.6. Results

- **Test Accuracy**: We have achieved **92.33%** (Meets the minimum requirement which is 91.5%)

- The model shows complete balanced performance across training and validation sets.

Part III: Optimizations

3.1 Enhanced Early Stopping

The EarlyStopping callback halts training if val_loss doesn't improve by at least 0.001 for 8 epochs, then restores the best model weights.

This prevents overfitting while optimizing performance.

3.2 Learning Rate Reduction

ReduceLROnPlateau callback dynamically halves the learning rate when val_loss stagnates for 3 epochs, with a lower bound of 1e-6, to refine convergence.

This balances aggressive training early with precision tuning later.

3.3. Optimized Results

- **Optimized Test Accuracy**: **92.83%**

- Accuracy increased +0.50% by using these optimization techniques.

Contribution summary

Team Member	Assignment Part	Contribution %
Jayant Som	Part 1: Visualizations, Code refactoring, NN model adjustments, accuracy increase, plots Part 2: Visualizations, Code refactoring, CNN model adjustments, accuracy increase, plots Part 3: Optimization adjustments	50%
Rithvik Illandula	Part 1: Preprocessing, details and statistics, NN architecture and model building Part 2: Preprocessing, details and statistics, CNN architecture and model building Part 3: Optimization testing, plots	50%

Acknowledgements

We would like to express our sincere gratitude to Prof. Nitin Kulkarni and TA. Xiaofeng Chen (CSE Dept, University at Buffalo) for their invaluable guidance, insightful feedback, and continuous support throughout this project. Their expertise and encouragement were instrumental in helping us understand the theoretical and practical aspects of neural networks, convolutional neural networks and optimization techniques.

We also extend our thanks to the University at Buffalo for providing the resources and infrastructure necessary to complete this work.

- Jayant Som
- Rithvik Illandula

References

https://www.datacamp.com/tutorial/pytorch-tutorial-building-a-simple-neural-network-from-scratch

https://www.digitalocean.com/community/tutorials/constructing-neural-networks-from-scratch

https://www.tensorflow.org/tutorials/images/cnn

https://towardsdatascience.com/building-a-convolutional-neural-network-cnns-from-scratch-3cfa453f9594/