ASSIGNMENT-3

Team members:

Vasavi devi Makinedi – vasavide (50544994)

Aditya Sri Vastav Lolla – alolla (50559685)

Introduction

This project focuses on using machine learning techniques to build a Neural Network (NN) and a Convolutional Neural Network (CNN) for handling different prediction tasks. The Neural Network model is implemented for prediction of income levels, where the CNN model classifies images from the Fashion MNIST dataset. This report details the processes involved from data preparation to model evaluation.

Income Dataset

The income dataset is a complete set of points, either for different demographic, occupational, or financial attributes of people. It mostly finds applications in exploratory data analysis, training machine learning models, and demographic studies. The dataset contains the following information about the individuals: age; work status; education level; marital status; occupation; relationship status; race; gender; capital gains and losses; hours worked per week; native country; and income category.

Detailed Overview of the Dataset:

Age: This attribute has stored the age of the people in the database. Age is one of the important demographic attributes. It impacts almost every aspect of human life, including their income level.

Workclass: It identifies the kind of job the occupants are involved in. There are private sector jobs, self-employed, government jobs, among others. Some entries are marked ?. These represent missing data.

fnlwgt: Default Weight — The weighting factor that the Census Bureau did to adjust for the sampling design, non-response and post-stratification. Education: This column gives the highest education level attained by the individual, like high school graduation, bachelors' degree, or some college education.

education.num: The column that makes the above column of educational attainment into numerical form, with increasing numbers signifying higher education.

Marital Status: This attribute confirms the marital status of individuals, which include never married, married, divorced, separated, and widowed.

Occupation: This attribute includes what the individuals are exactly engaged in: managerial, professional, or technical. The '?' entries represent missing data.

Relationship: It gives the relationship of the individuals with their respective houses, specifying them as spouse, child, not-in-family, and so on. It is an attributeous column that describes the racial background of a person. It holds fast to entries like White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, and so on.

Sex: This attribute tells the sex of each person, being Male or Female. Capital Gain: Capital gains are increases in value from the sale of assets or investments; this column represents that very fact.

Capital Loss: Capital losses are losses incurred by selling assets or investments; that is what this column represents.

Hours per Week: The number of hours a person spends working in a week. Native Country: This column represents one's country of origin, nearly the bulk being from the United States, although members are from other countries as well.

Income: In this case, it is represented as a target variable. It defines whether a person earns above or below $50,000 per year, such as <=50K or >50K.

Nature of Dataset:

This dataset includes both numerical and categorical data, each having a distinct purpose in processing the model.

Age: This numerical attribute quantifies the age of individuals, providing a measurable insight into the demographic profile of the dataset.

Hours per Week: Indicates how many hours individuals are employed each week, a crucial factor that often correlates with their income levels.

Education Num: This metric quantifies the years of education that an individual has completed, typically linked to their potential for higher earnings.

These numerical variables can be directly incorporated into calculations and statistical models. They are instrumental in identifying patterns and relationships through mathematical and statistical analyses, such as determining how they correlate with income levels.

Marital Status: This category includes labels such as "Married", "Single", "Divorced", etc., each of which may have an impact on an individual's earning capacity.

Occupation: This describes the specific role or job type an individual has, with examples like "Tech-support", "Craft-repair", "Exec-managerial", etc.

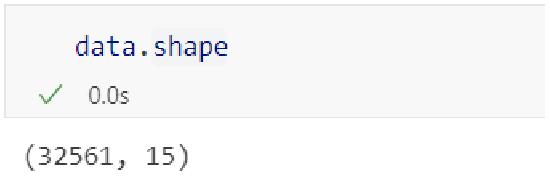
Gender: Common categories in this field are "Male" and "Female".

Race: This attribute is classified into various groups, such as "White", "Black", "Asian", "Hispanic", etc.

Native Country: Indicates the individual's country of birth.

Categorical data typically need to be converted into a numerical format using methods like one-hot encoding or label encoding before they are viable for use in machine learning algorithms. This conversion is essential since these algorithms predominantly operate on numerical inputs.

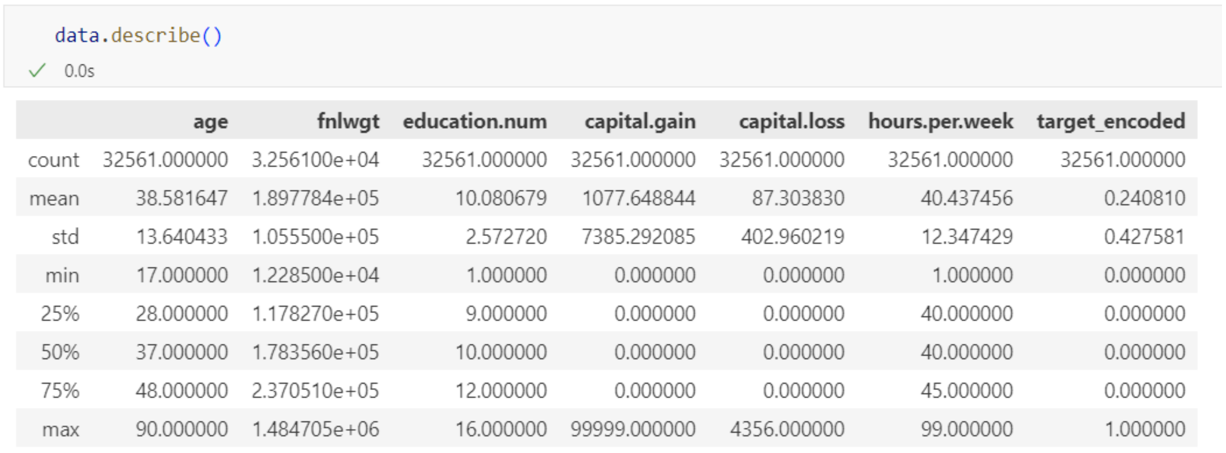
To know the entries and the number of variables, we use **.shape** attribute.



```
data.shape
✓ 0.0s
(32561, 15)
```

The dataset comprises of 32561 entries and 15 variables.

To provide the main statistics of the dataset, we use the **.describe( )** attribute from pandas which gives the summary of data like mean, min, min, central tendency.



```
data.describe()
✓ 0.0s
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	target_encoded
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456	0.240810
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429	0.427581
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	0.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000	0.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000	0.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000	0.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000	1.000000

Statistical Breakdown

- **Age**
  - **Count**: 32,561 entries, indicating complete data with no missing values.
  - **Mean**: The average age is approximately 38.59 years.
  - **Standard Deviation**: The age data has a variation of about 13.64 years around the mean.
  - **Minimum**: The youngest individual is 17 years old.
  - **25%**: 25% of the individuals are 28 years old or younger.
  - **Median (50%)**: 37 years is the median age, which indicates that half of the individuals are younger than 37 and the other half older.
  - **75%**: 75% of the individuals are 48 years old or younger.
  - **Maximum**: The oldest individual is 90 years old.
- **Final Weight (fnlwgt)**
  - This attribute estimates the number of individuals that each census entry corresponds to, reflecting the demographic characteristics within the population.

    - **Mean**: The average final weight is approximately 1.897 million.
    - **Minimum to Maximum**: Ranges from about 12,285 to 1,484,705, indicating significant variation in the representation weight of entries.
- **Education Number (education.num)**
    - This is likely a proxy for years of education.
    - **Mean**: The average years of education completed is about 10.08.
    - **Minimum to Maximum**: Ranges from 1 to 16, suggesting education levels from no formal education to advanced degrees.
- **Capital Gain and Capital Loss**
    - These attributes represent income from investments outside of wages/salary.
    - **Capital Gain**:
        - **Mean**: Average capital gain is approximately 1,077.65.
        - **Maximum**: The highest capital gain recorded is 99,999.
    - **Capital Loss**:
        - **Mean**: Average capital loss is around 87.30.
        - **Maximum**: The highest capital loss recorded is 4,356.
- **Hours per Week (hours.per.week)**
    - This represents the total working hours per week.
    - **Mean**: On average, individuals work about 40.44 hours per week.
    - **Minimum to Maximum**: Ranges from 1 to 99 hours, showcasing a wide variety of part-time to multiple-job holdings.
- **Target Encoded**
    - This is likely a binary encoded target variable for income, perhaps representing whether income exceeds a certain threshold (e.g., $50,000).
    - **Mean**: On average, about 24.08% of the entries exceed this income threshold.

Libraries Used:

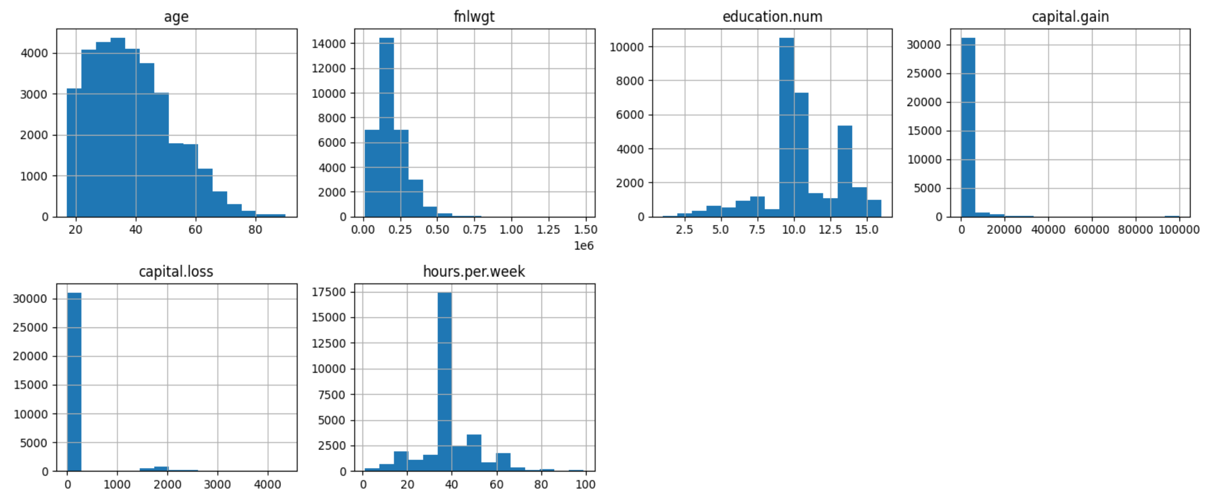
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.metrics import confusion_matrix

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D,
MaxPooling2D, Flatten, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,
LearningRateScheduler
from tensorflow.keras.regularizers import l2


from keras.datasets import fashion_mnist
```

Visualizations:

Histograms

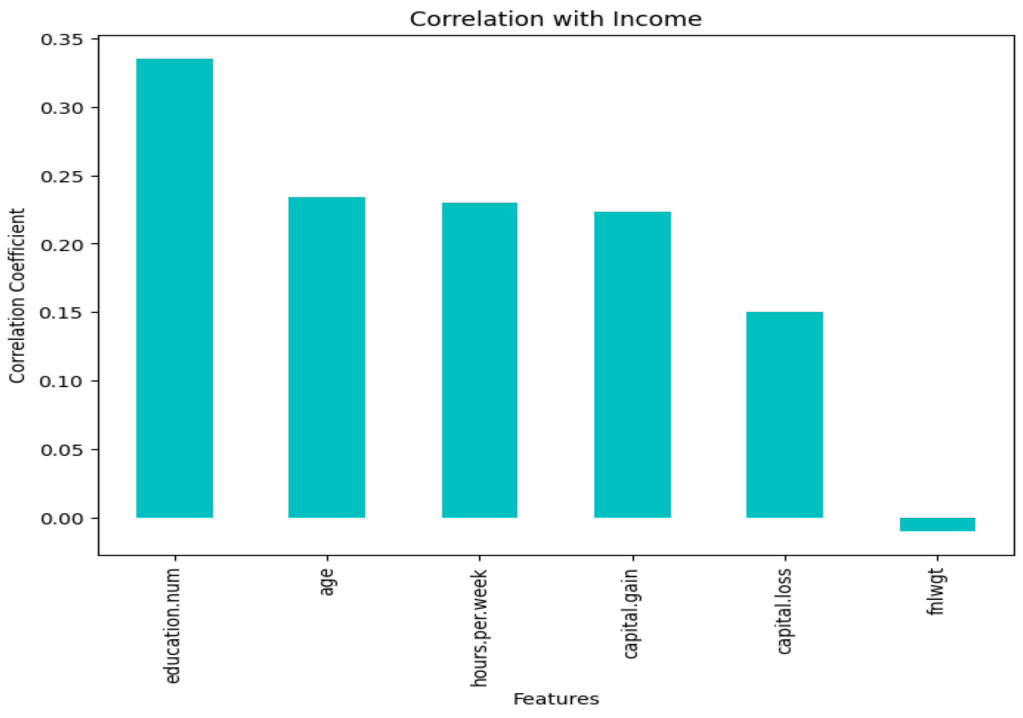
- **Age Distribution**: This histogram shows a bi-modal distribution of ages, peaking around 20-30 years and again at around 50 years, suggesting a significant representation of young adults and middle-aged individuals in the dataset.
- **Final Weight Distribution**: The distribution of final weights (fnlwgt), which represents demographic weights in the census data, is right-skewed, indicating that most individuals have a relatively low weight, with few having very high weights.
- **Education Number Distribution**: This graph displays a mode around 9-10 years of education, likely indicating a high prevalence of high school graduates in the dataset.
- **Capital Gain Distribution**: Capital gain values are mostly clustered at zero, indicating that most individuals do not have income from capital gains, with a few exceptions reaching as high as 100,000.

- **Capital Loss Distribution**: Similar to capital gains, capital loss values are predominantly zero, showing that losses are not common among the dataset's individuals.
- **Hours Per Week Distribution**: This histogram is sharply peaked around 40 hours per week, typical of full-time employment, with fewer individuals working significantly more or fewer hours.



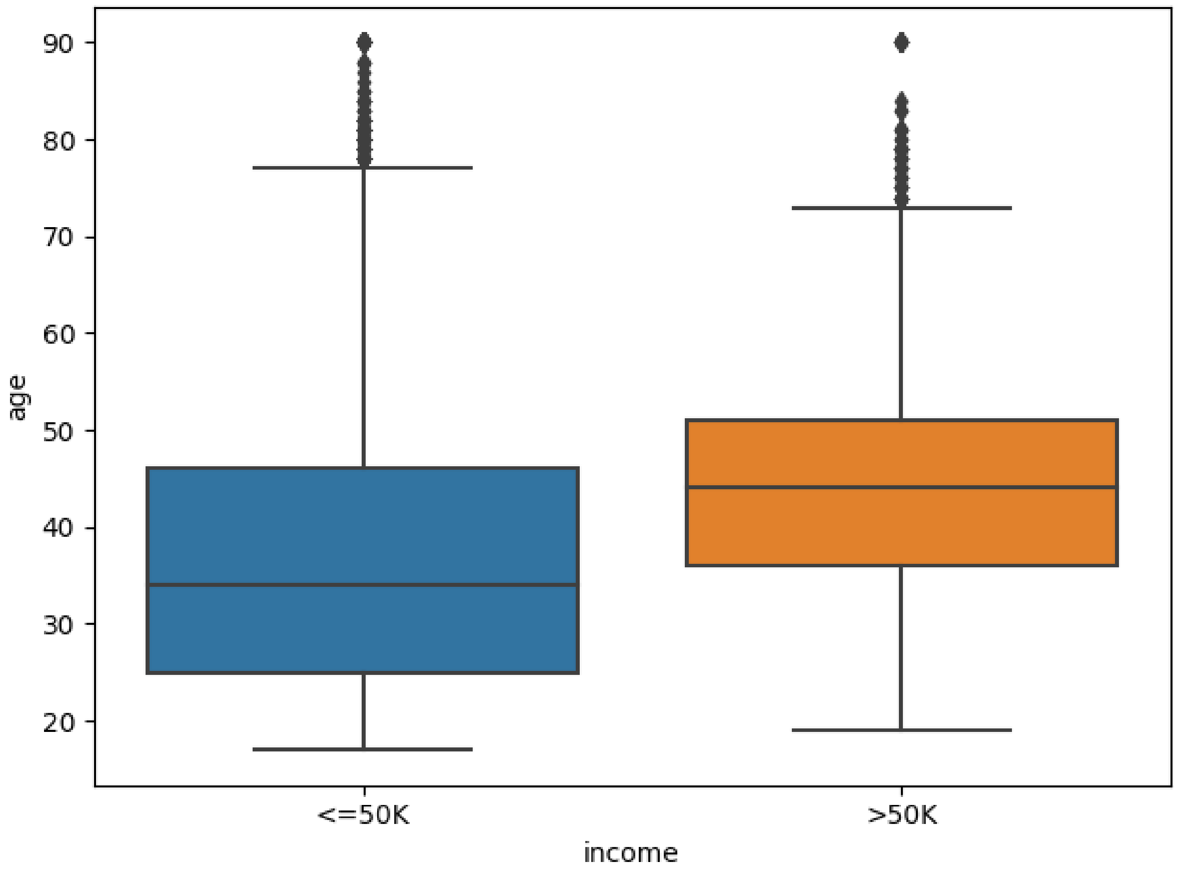
Correlation graph

This graph displays the strength of linear associations between several features and the target variable. 'Education number' shows the highest correlation, followed closely by 'age' and 'hours per week', indicating strong relationships with the target variable “Income”. 'Capital gain' also shows a significant correlation, whereas 'capital loss' and 'fnlwgt' exhibit much weaker correlations. This visualization helps in identifying which features might be most predictive of the target variable in subsequent modelling efforts.

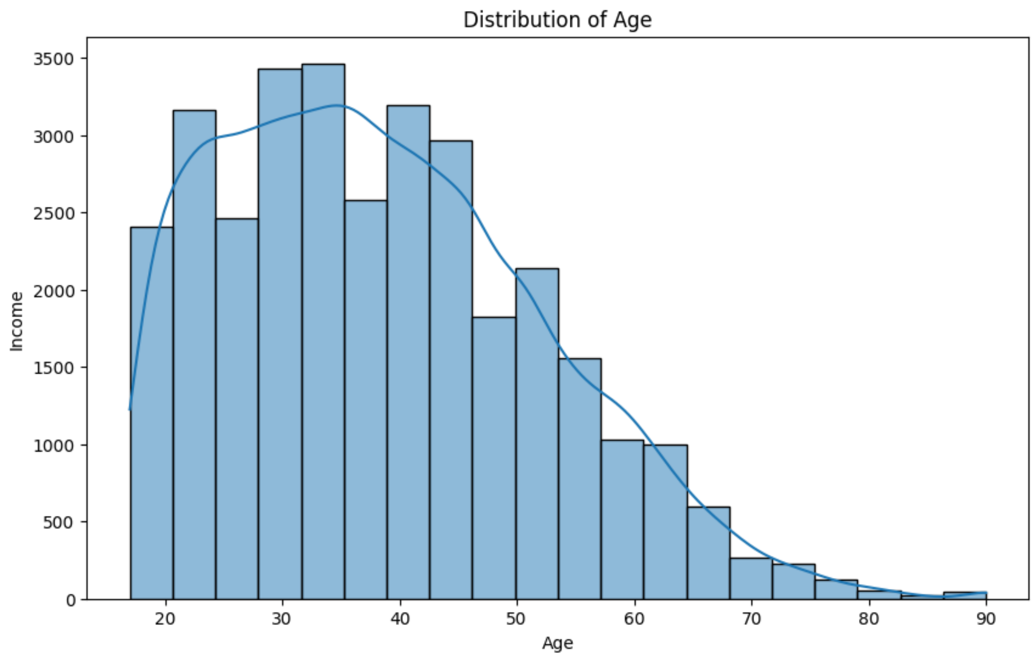


Box plot

This boxplot showcases how age distribution varies with income levels. For individuals earning "<= $50K", the age distribution is concentrated in a younger demographic with a narrower interquartile range, suggesting that lower incomes are more common among younger adults. Conversely, the boxplot for those earning "> $50K" features a higher median age and a wider interquartile range, indicating that higher incomes are typically found among older age groups. This graph provides insight into the relationship between age and income, suggesting that income potential increases with age.



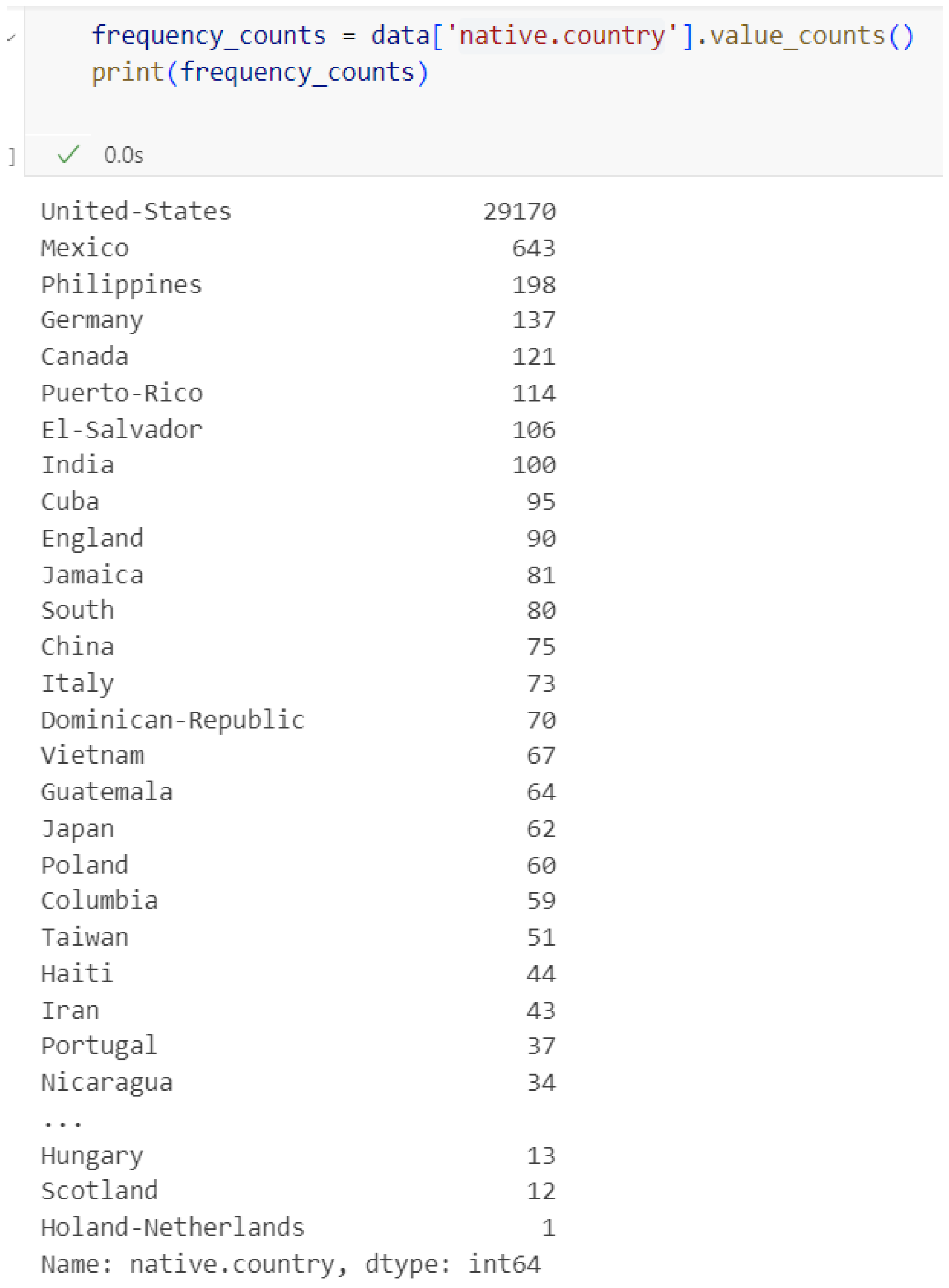
This histogram visually represents the frequency of different age groups within the dataset, showing a significant concentration of individuals in their late 20s to early 50s. The distribution peaks in the mid-30s, indicating a large proportion of the population within this age range, and gradually declines as age increases, reflecting fewer older adults within the dataset.



Data Cleaning and Feature Engineering

Data Cleaning

- **Dropping Redundant and Non-Informative Features**: The columns 'education' and ‘education.num’ both represent the same data. Hence, the column ‘education’ is dropped.
- The column 'native.country' has most of the entries from United States. Hence, this column is removed.



```
frequency_counts = data['native.country'].value_counts()
print(frequency_counts)
```

```
United-States              29170
Mexico                       643
Philippines                  198
Germany                      137
Canada                       121
Puerto-Rico                  114
El-Salvador                  106
India                        100
Cuba                          95
England                       90
Jamaica                       81
South                         80
China                         75
Italy                         73
Dominican-Republic            70
Vietnam                       67
Guatemala                     64
Japan                         62
Poland                        60
Columbia                      59
Taiwan                        51
Haiti                         44
Iran                          43
Portugal                      37
Nicaragua                     34
...
Hungary                       13
Scotland                      12
Holand-Netherlands             1
Name: native.country, dtype: int64
```

- **Handling Missing Values**: The dataset is inspected for missing values (data.isnull().sum()), and placeholders such as '?' are replaced with NaN to standardize missing data representation. This step is crucial for the effective application of imputation techniques.
- **Visualizing Missing Data**: The missingno library is used to visualize the distribution and pattern of missing data, which aids in deciding how to handle these missing values effectively.

Data Transformation

- **Encoding of the Target Variable**: The 'income' column is transformed into a binary format suitable for classification models, facilitating model interpretation and evaluation.
- **Correlation Analysis**: A correlation matrix is computed to assess the relationship between the transformed target variable and other features, guiding feature selection and understanding feature importance.

Numerical and Categorical Feature Processing

- **Pipelines for Numerical and Categorical Data**: Separate pipelines are created for numerical and categorical data:
    - **Numerical Data Pipeline**: Includes imputation of missing values with the mean and scaling of features using StandardScaler. Scaling numerical features is essential as it normalizes data, ensuring that no single feature disproportionately influences the model's outcome.
    - **Categorical Data Pipeline**: Implements imputation of missing values with the most frequent category and one-hot encoding of categorical variables. This transformation converts categorical variables into a machine-readable numeric format, expanding the applicability of machine learning algorithms that require numerical input.

Integration into a Unified Preprocessing Framework

- **Column Transformer**: This tool integrates both pipelines into a single processing step, applying appropriate transformations to each data type (numerical or categorical). This setup not only streamlines the preprocessing workflow but also helps prevent data leakage by ensuring that transformations fit on the training data are applied consistently to any new data.

Architecture structure of Neural Network

Input Layer

- The network starts with an input layer designed to receive data, matching the number of features processed from the training dataset (X_train.shape[1]). This configuration ensures complete feature representation within the network.

First Hidden Layer

- **Neurons**: Contains 64 neurons, setting the layer's capacity to model complex data representations, though it requires careful management to prevent overfitting.
- **Activation Function**: Utilizes the ReLU (Rectified Linear Unit) activation to introduce non-linearity, facilitating the learning of complex data patterns.
- **Regularization**: Employs L2 regularization (lambda = 0.01) on the neuron weights to mitigate overfitting by penalizing large weights.
- **Dropout**: Incorporates a dropout rate of 0.3, randomly setting 30% of the neuron outputs to zero during training, which helps in promoting model generalization by learning redundant representations.

Second Hidden Layer

- Mirrors the structure of the first hidden layer with 64 neurons using ReLU activation. A dropout rate of 0.3 is applied, consistent with the approach to manage overfitting and ensure robust feature learning.

Output Layer

- **Neurons**: Comprises a single neuron, reflecting the binary classification nature of the task (e.g., determining if income exceeds $50K).
- **Activation Function**: The sigmoid activation function narrows the output to a range from 0 to 1, making it ideal for binary classification tasks as it offers a probabilistic interpretation of the output.

Model Compilation

- **Optimizer**: The Adam optimizer with a learning rate of 0.0005 is chosen for its efficiency and adaptability across different neural architectures.
- **Loss Function**: Uses 'binary_crossentropy', ideal for binary classification where the outcomes are labelled as 0 or 1.
- **Metrics**: Monitors 'accuracy' to evaluate the proportion of correctly predicted instances during the model's training and evaluation phases.

Training Process

- **Epochs**: The model undergoes training for 50 epochs, iterating through the complete dataset multiple times to optimize learning.
- **Batch Size**: Batch size is 32, which determines the number of samples processed before the model's parameters are updated.

- **Validation Split**: 20% of the training data is reserved as a validation set, used to prevent overfitting and fine-tune the model parameters based on performance feedback.

Model Evaluation

- After training, the model's effectiveness is tested on the test dataset, with the accuracy percentage of 85.61. This step confirms the model's ability to generalize beyond the training data.

Accuracy and Loss graphs

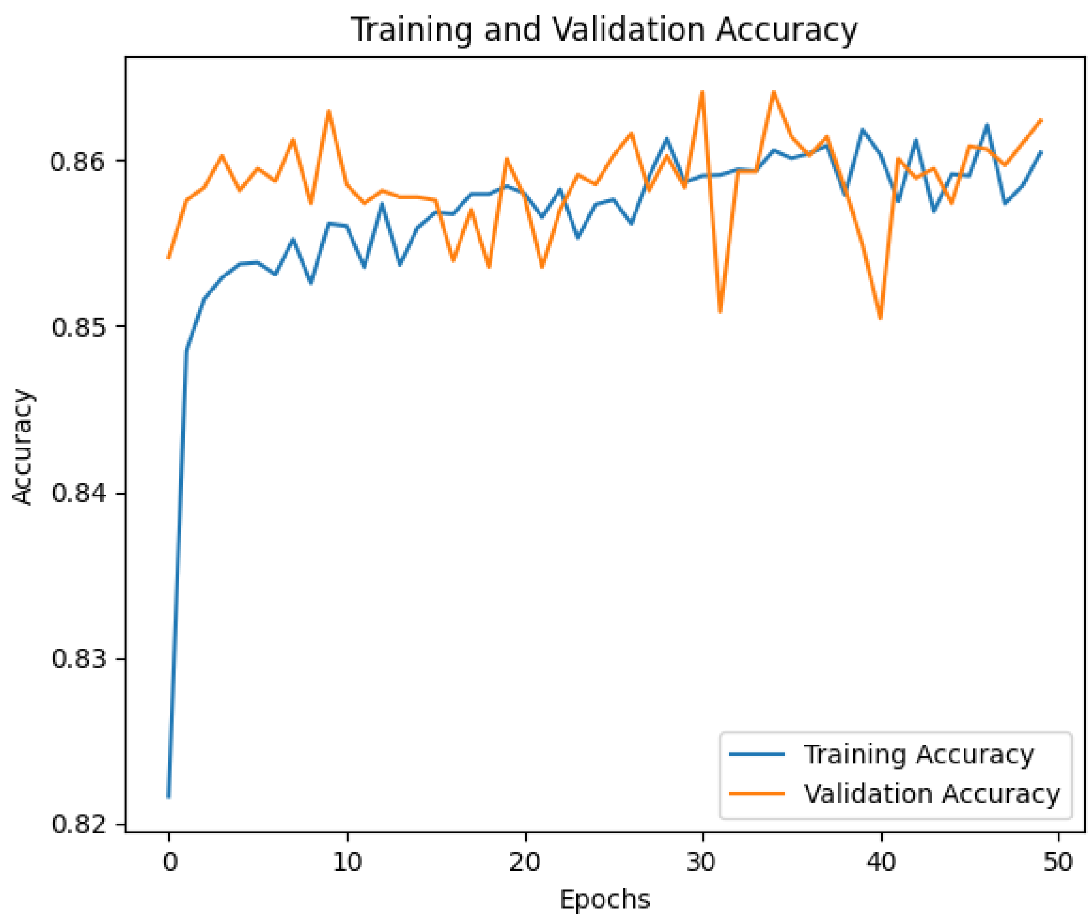
The below graphs visually compare the training and validation metrics for accuracy and loss over a series of training epochs, providing insight into the neural network's performance during the training process.

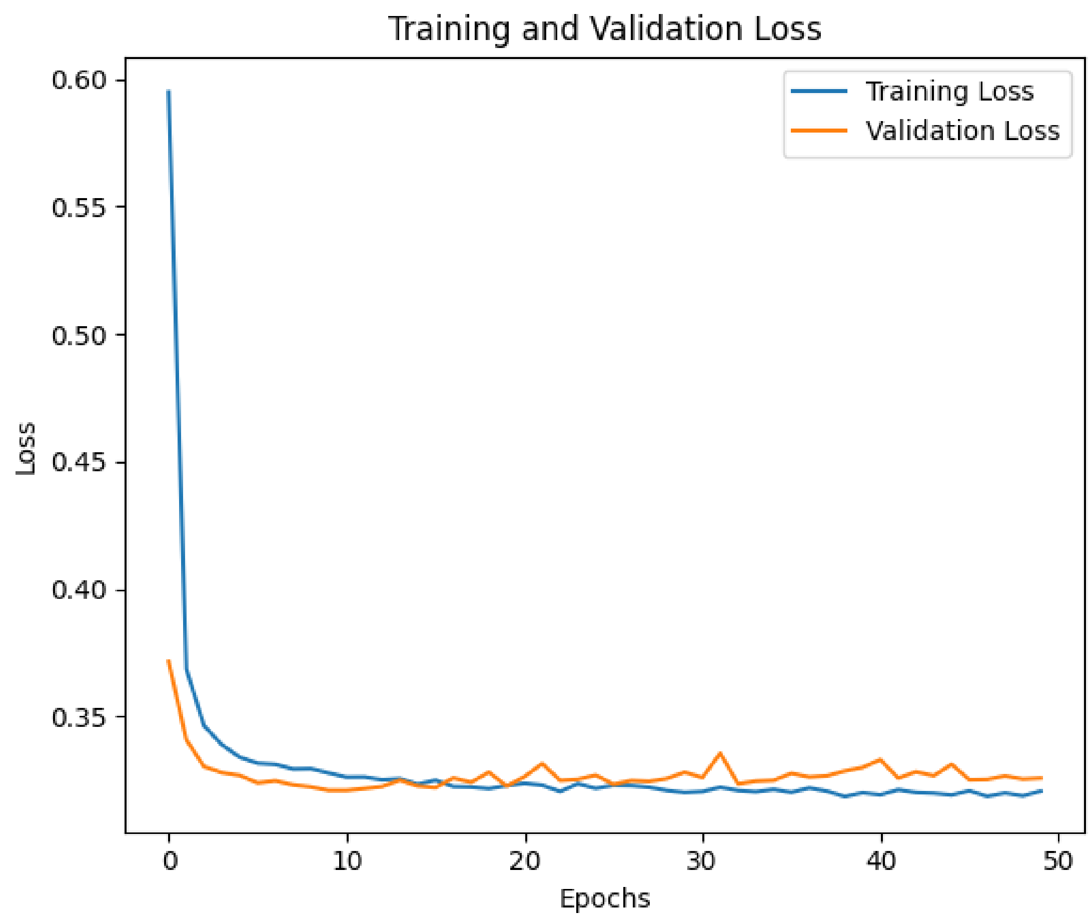
Training and Validation Accuracy Graph

This graph displays the model's accuracy during training and validation (blue line and orange line respectively) across 50 epochs. Initially, there is a sharp increase in training accuracy, indicating rapid learning in the early stages. The training accuracy then stabilizes and fluctuates slightly, maintaining high levels throughout the remaining epochs. In contrast, the validation accuracy experiences some volatility, generally aligning closely with the training accuracy but with more frequent fluctuations. This pattern suggests a good generalization by the model, although the slight divergence and fluctuations in validation accuracy may indicate minor overfitting or model sensitivity to the validation data specifics.

Training and Validation Loss Graph

The loss graph shows the model's training (blue line) and validation (orange line) loss over the same 50 epochs. Both loss metrics decrease sharply initially, reflecting significant improvements in the model's predictive accuracy early in training. As the epochs continue, both training and validation losses plateau and closely align, demonstrating a consistent reduction in error rates as the model learns. The convergence of training and validation loss at low levels towards the end of the training indicates that the model is not overfitting significantly, as the validation loss remains close to the training loss, suggesting effective learning and generalization capabilities.


Training and Validation Accuracy
Accuracy
0.82
0.83
0.84
0.85
0.86
0
10
20
30
40
50
Epochs
Training Accuracy
Validation Accuracy


Training and Validation Loss
Loss
0.35
0.40
0.45
0.50
0.55
0.60
0
10
20
30
40
50
Epochs
Training Loss
Validation Loss

These graphs are crucial for assessing the model's performance, indicating effective learning and the ability to generalize from training data to unseen validation data without significant overfitting. The close tracking of validation metrics with training metrics throughout the training process is a positive indicator of the model's robustness.

PART 2 – Fashion MNIST DATASET

Overview:

The Fashion-MNIST dataset is specifically designed for developing and training machine learning models on fashion product images, making it a more interesting and challenging alternative to the traditional handwritten digit MNIST dataset. It comprises 70,000 grayscale images, each of 28x28 pixel resolution, spread across 10 fashion categories such as shirts, dresses, shoes, and bags. This dataset is a benchmark for evaluating machine learning algorithms, particularly in computer vision.

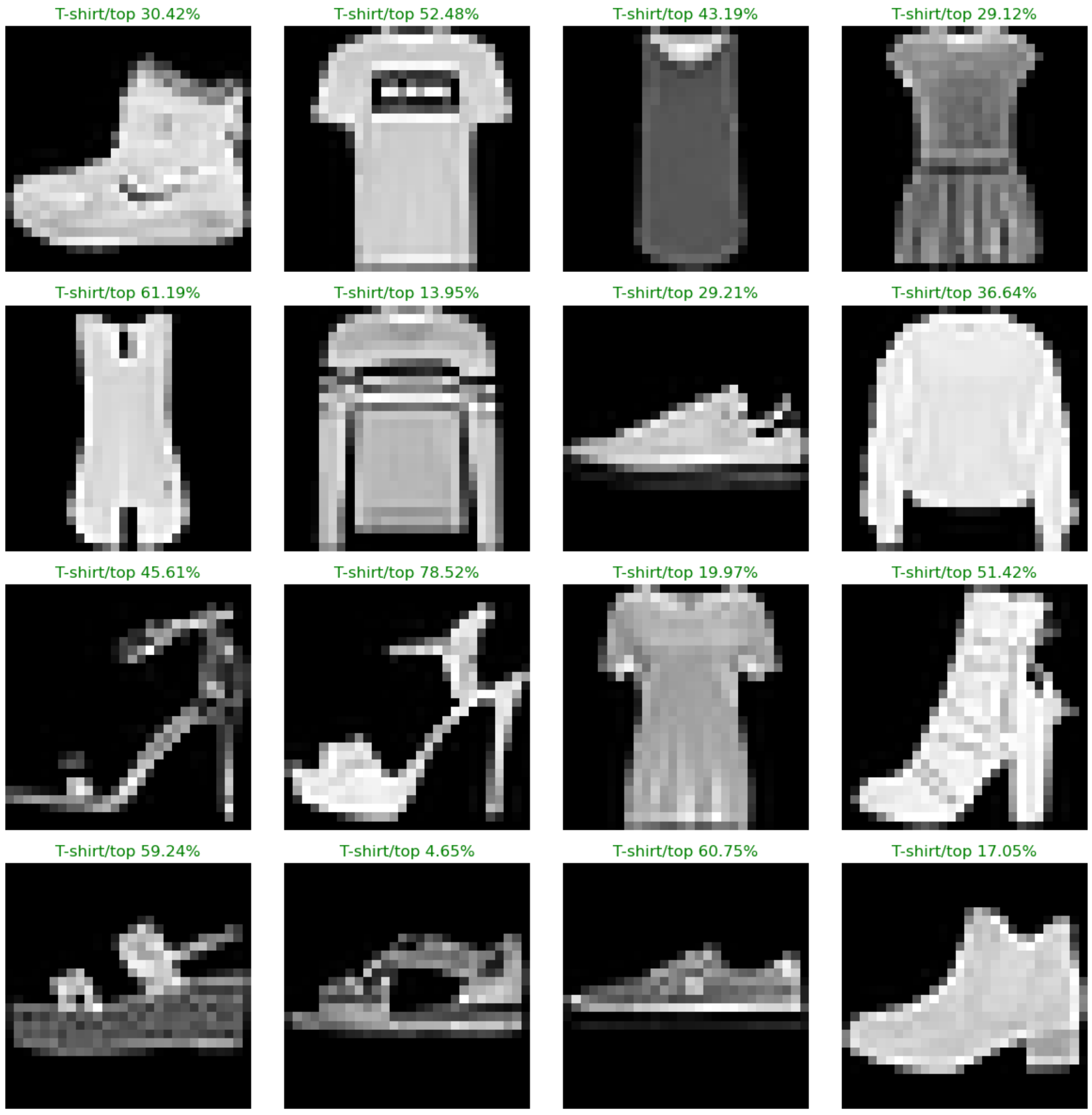
Libraries:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import make_column_transformer
from sklearn.metrics import confusion_matrix

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D, Flatten, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRateScheduler
from tensorflow.keras.regularizers import l2


from keras.datasets import fashion_mnist
```

Data Visualization:

Let's whip up a grid of sample pics from our dataset. We'll show off the different clothes - tees pants, kicks - each with its own label. This'll give us a good look at what we're dealing with. Check it out! Here's a peek at the Fashion-MNIST dataset. It's got all sorts of threads - tees, pants, dresses, shoes. Each pic is a tiny 28x28 grayscale image. This grid gives you a taste of what's in the dataset. It's like a little fashion show of all the different styles we're working with.

Here the styling Consist of



Architecture structure of CNN

Convolutional Neural Network (CNN) architecture is structured to handle image classification tasks effectively, with multiple convolutional, pooling, and dense layers to process and classify images. Here's a detailed breakdown of each component of the CNN architecture:

Convolutional Layers

- **First Convolutional Layer**:
  - **Filters**: 32
  - **Kernel Size**: 3x3
  - **Activation Function**: ReLU (Rectified Linear Unit)
  - **Input Shape**: (28, 28, 1) - This specifies that each input image has dimensions 28x28 with one color channel (grayscale).
  - **Batch Normalization**: Applied after this layer to normalize the activations and speed up training.
- **Pooling Layer**:
  - **Type**: MaxPooling
  - **Pool Size**: 2x2
  - Reduces the spatial dimensions (height and width) by half, which helps in making the representation smaller and more manageable.
- **Dropout**:
  - **Rate**: 25%
  - Reduces overfitting by randomly setting a fraction of the input units to 0 at each update during training time.
- **Second Convolutional Layer**:
  - **Filters**: 64
  - Uses the same kernel size and activation function.
  - **Batch Normalization**: Again applied to normalize the activations.
- **Third Convolutional Layer**:
  - **Filters**: 64
  - Continues with the same configuration, strengthening the network's ability to recognize complex patterns in the image data.

Flattening Layer

- Transforms the 3D feature maps into 1D feature vectors, making them suitable for input into the dense layers for classification.

Dense Layers

- **First Dense Layer**:
  - **Units**: 250
  - **Activation Function**: ReLU
- **Second Dense Layer**:
  - **Units**: 125
  - **Activation Function**: ReLU
- **Third Dense Layer**:
  - **Units**: 60
  - **Activation Function**: ReLU

Output Layer

- **Units**: 10 (corresponding to the 10 classes of digits)
- **Activation Function**: Softmax, which is used for multi-class classification. It outputs the probabilities of each class, with the target class having the highest probability.

Dropout Layers

- Integrated before each dense layer (excluding the output layer) with a dropout rate of 25% to prevent overfitting by reducing the complexity of the model.

Model Performance

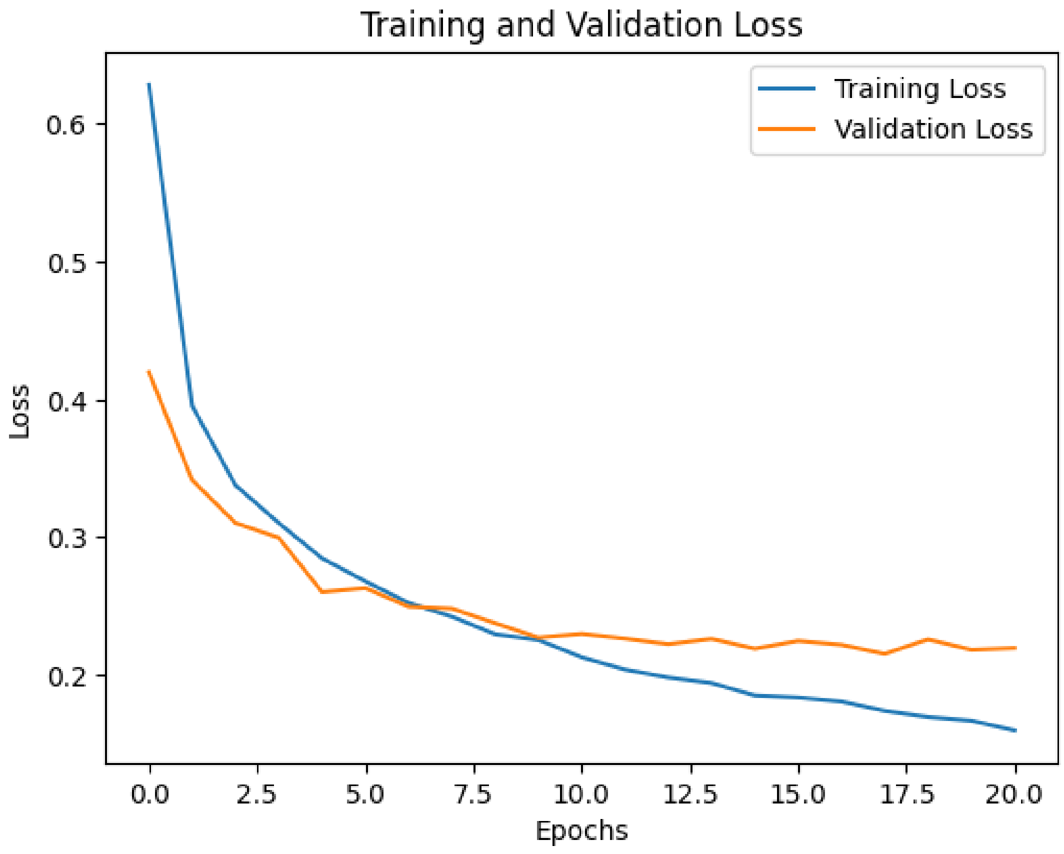
- The model achieves a notable accuracy of 92.02%, demonstrating its effectiveness in accurately classifying images. This high level of accuracy indicates robust learning and generalization capabilities across diverse image inputs.

Training and Validation Loss Graph

This graph shows a sharp decrease in both training and validation loss during the initial epochs, indicating that the model is quickly learning from the training data. As the epochs progress, both lines begin to plateau, suggesting diminishing returns from further training on reducing loss.

- **Observations**:
  - **Convergence**: The training and validation loss lines are converging, which is a good indicator that the model is not overfitting significantly. The validation loss closely follows the training loss, showing that the model generalizes well to new data.

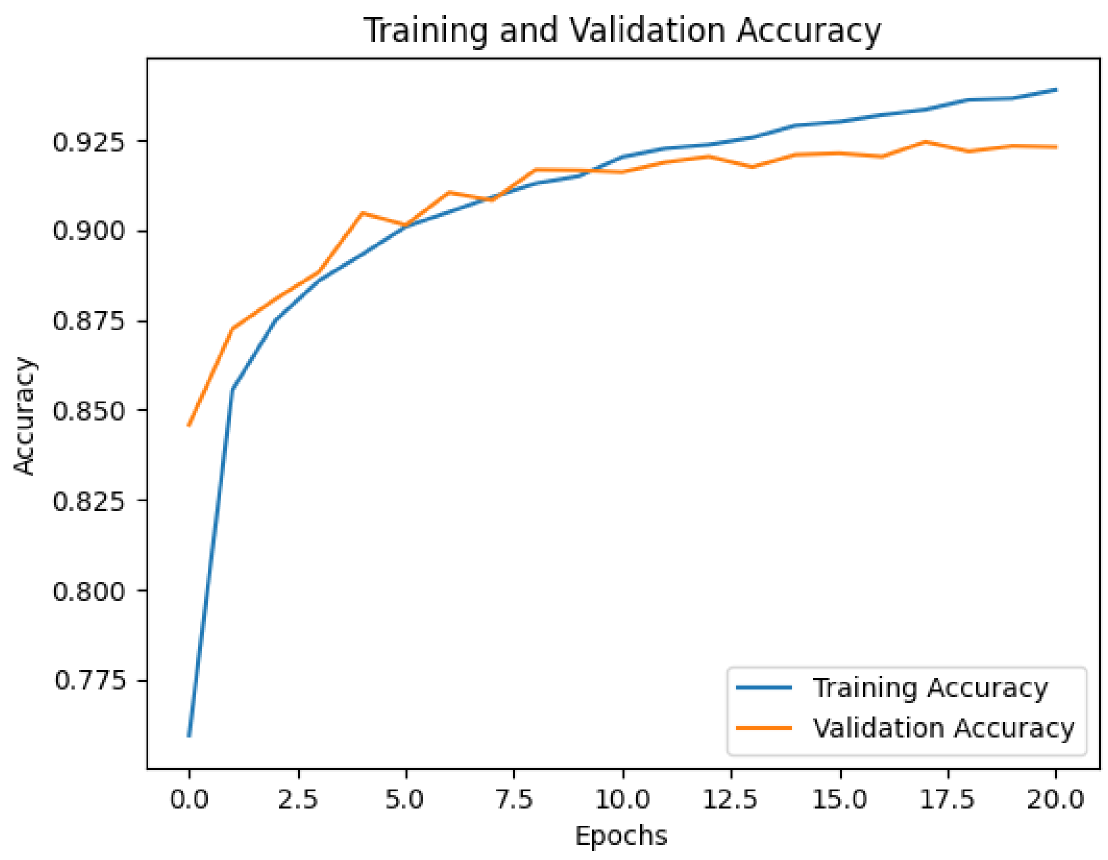
- **Stability**: The smooth and steady decline in loss values reflects stable learning parameters and an effective learning rate.



Training and Validation Accuracy Graph

This graph tracks the accuracy of the model on both the training set and the validation set across the same 20 epochs. Both accuracy metrics improve significantly in the early epochs and continue to rise gradually, eventually plateauing toward the end of the training period.

- **Observations**:
  - **High Accuracy Levels**: Both training and validation accuracies are high, with training accuracy slightly higher than validation accuracy, which is typical in machine learning scenarios.
  - **Narrowing Gap**: The gap between training and validation accuracy remains narrow throughout the training process, which reinforces the model's ability to generalize without memorizing the training data.



Interpretation and Implications

- **Effective Learning**: The trends in both graphs indicate effective learning dynamics, where the model rapidly absorbs the training data's patterns and also proves capable of generalizing these patterns to unseen data (validation set).
- **Model Configuration and Training**: The depicted performance metrics suggest that the current configuration of the network, including its architecture, learning rate, and regularization techniques (potentially dropout or weight decay), is well-tuned for the given task.
- **Potential for Further Optimization**: While the performance is strong, the slight divergence between training and validation metrics toward the last epochs might suggest room for further optimization, possibly through early stopping, adjustments in the learning rate, or more advanced regularization techniques to align the validation performance more closely with the training performance.

Contribution

Person	Contribution	Contribution %
Aditya Sri Vastav Lolla	Worked on CNN model for Fashion MNIST dataset, improving Accuracy and Report	50
Vasavi devi Makinedi	Worked on Neural Network model for income dataset, improving accuracy for part 1 and Report	50

References

https://stackoverflow.com/questions/53837545/plotting-mnist-samples

https://github.com/RUPAKparajuli07/Fashion-MNIST-in-deep-learning

https://github.com/junaidaliop/pytorch-fashionMNIST-tutorial

https://machinelearningmastery.com/data-visualization-in-python-with-matplotlib-seaborn-and-bokeh/

How to Analyze Income Dataset Using Pandas and Visualization Libraries | by Nutan | Medium

Pyplot tutorial — Matplotlib 3.9.1 documentation