

CytoAutoCluster

Importing Header files

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.semi_supervised import LabelPropagation
from sklearn.metrics import silhouette_score
from sklearn.manifold import TSNE
```

```
In [ ]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Displaying the data

```
In [ ]: import pandas as pd

# Provide the URL of the file
url = '/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv'

# Load the dataset
df = pd.read_csv(url)

# Check the first few rows of the dataset
print(df.head())
```

	Event	Time	Cell_length		DNA1	DNA2	CD45RA	CD133	\
0	1	2693.0	22	4.391057	4.617262	0.162691	-0.029585		
1	2	3736.0	35	4.340481	4.816692	0.701349	-0.038280		
2	3	7015.0	32	3.838727	4.386369	0.603568	-0.032216		
3	4	7099.0	29	4.255806	4.830048	0.433747	-0.027611		
4	5	7700.0	25	3.976909	4.506433	-0.008809	-0.030297		

	CD19	CD22	CD11b	...	CD117	CD49d	HLA-DR	CD64	\
0	-0.006696	0.066388	-0.009184	...	0.053050	0.853505	1.664480	-0.005376	
1	-0.016654	0.074409	0.808031	...	0.089660	0.197818	0.491592	0.144814	
2	0.073855	-0.042977	-0.001881	...	0.046222	2.586670	1.308337	-0.010961	
3	-0.017661	-0.044072	0.733698	...	0.066470	1.338669	0.140523	-0.013449	
4	0.080423	0.495791	1.107627	...	-0.006223	0.180924	0.197332	0.076167	

	CD41	Viability	file_number	event_number	label	individual
0	-0.001961	0.648429	3.627711	307	1.0	1
1	0.868014	0.561384	3.627711	545	1.0	1
2	-0.010413	0.643337	3.627711	1726	1.0	1
3	-0.026039	-0.026523	3.627711	1766	1.0	1
4	-0.040488	0.283287	3.627711	2031	1.0	1

[5 rows x 42 columns]

In []: df

Out[]:

	Event	Time	Cell_length	DNA1	DNA2	CD45RA	CD133	\
0	1	2693.00	22	4.391057	4.617262	0.162691	-0.029585	
1	2	3736.00	35	4.340481	4.816692	0.701349	-0.038280	
2	3	7015.00	32	3.838727	4.386369	0.603568	-0.032216	
3	4	7099.00	29	4.255806	4.830048	0.433747	-0.027611	
4	5	7700.00	25	3.976909	4.506433	-0.008809	-0.030297	
...	
265622	265623	707951.44	41	6.826629	7.133022	1.474081	-0.01917	
265623	265624	708145.44	45	6.787791	7.154026	0.116755	-0.05621	
265624	265625	708398.44	41	6.889866	7.141219	0.684921	-0.00626	
265625	265626	708585.44	39	6.865218	7.144353	0.288761	-0.01131	
265626	265627	709122.44	41	6.887820	7.127359	0.360753	0.12860	

265627 rows x 42 columns

In []: df.columns

```
Out[ ]: Index(['Event', 'Time', 'Cell_length', 'DNA1', 'DNA2', 'CD45RA', 'CD133',
              'CD19', 'CD22', 'CD11b', 'CD4', 'CD8', 'CD34', 'Flt3', 'CD20', 'CXCR
              4',
              'CD235ab', 'CD45', 'CD123', 'CD321', 'CD14', 'CD33', 'CD47', 'CD11
              c',
              'CD7', 'CD15', 'CD16', 'CD44', 'CD38', 'CD13', 'CD3', 'CD61', 'CD11
              7',
              'CD49d', 'HLA-DR', 'CD64', 'CD41', 'Viability', 'file_number',
              'event_number', 'label', 'individual'],
              dtype='object')
```

```
In [ ]: df['Viability']
```

```
Out[ ]:
```

	Viability
--	-----------

0	0.648429
1	0.561384
2	0.643337
3	-0.026523
4	0.283287
...	...
265622	0.236957
265623	-0.003500
265624	0.107206
265625	0.620872
265626	0.310466

265627 rows × 1 columns

dtype: float64

```
In [ ]: df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265627 entries, 0 to 265626
Data columns (total 42 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Event                 265627 non-null  int64
1   Time                 265627 non-null  float64
2   Cell_length          265627 non-null  int64
3   DNA1                 265627 non-null  float64
4   DNA2                 265627 non-null  float64
5   CD45RA               265627 non-null  float64
6   CD133                265627 non-null  float64
7   CD19                 265627 non-null  float64
8   CD22                 265627 non-null  float64
9   CD11b                265627 non-null  float64
10  CD4                  265627 non-null  float64
11  CD8                  265627 non-null  float64
12  CD34                 265627 non-null  float64
13  Flt3                 265627 non-null  float64
14  CD20                 265627 non-null  float64
15  CXCR4                265627 non-null  float64
16  CD235ab              265627 non-null  float64
17  CD45                 265627 non-null  float64
18  CD123                265627 non-null  float64
19  CD321                265627 non-null  float64
20  CD14                 265627 non-null  float64
21  CD33                 265627 non-null  float64
22  CD47                 265627 non-null  float64
23  CD11c                265627 non-null  float64
24  CD7                  265627 non-null  float64
25  CD15                 265627 non-null  float64
26  CD16                 265627 non-null  float64
27  CD44                 265627 non-null  float64
28  CD38                 265627 non-null  float64
29  CD13                 265627 non-null  float64
30  CD3                  265627 non-null  float64
31  CD61                 265627 non-null  float64
32  CD117                265627 non-null  float64
33  CD49d                265627 non-null  float64
34  HLA-DR               265627 non-null  float64
35  CD64                 265627 non-null  float64
36  CD41                 265627 non-null  float64
37  Viability             265627 non-null  float64
38  file_number           265627 non-null  float64
39  event_number          265627 non-null  int64
40  label                 104184 non-null  float64
41  individual            265627 non-null  int64
dtypes: float64(38), int64(4)
memory usage: 85.1 MB

```

```
In [ ]: df.describe()
```

Out[]:

	Event	Time	Cell_length	DNA1	D
count	265627.000000	265627.000000	265627.000000	265627.000000	265627.000000
mean	132814.000000	272948.345014	34.450572	4.606956	5.190000
std	76680.054314	171220.139430	11.446694	1.312831	1.150000
min	1.000000	1.000000	10.000000	2.786488	2.230000
25%	66407.500000	120196.000000	26.000000	3.700023	4.400000
50%	132814.000000	253276.000000	33.000000	4.022127	4.690000
75%	199220.500000	424502.500000	41.000000	6.353313	6.760000
max	265627.000000	709122.440000	65.000000	7.001489	7.470000

8 rows × 42 columns

Finding the column containg null values

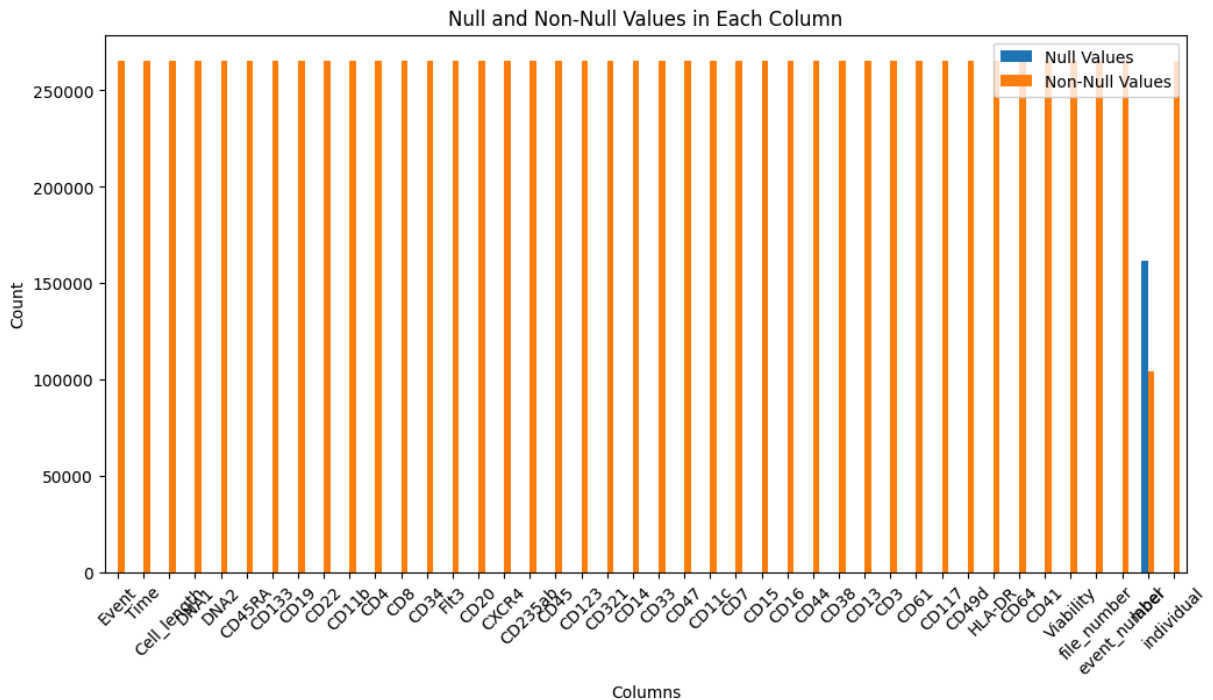
```
In [ ]: null_count = df.isnull().sum()
print(null_count)
```

Event	0
Time	0
Cell_length	0
DNA1	0
DNA2	0
CD45RA	0
CD133	0
CD19	0
CD22	0
CD11b	0
CD4	0
CD8	0
CD34	0
Flt3	0
CD20	0
CXCR4	0
CD235ab	0
CD45	0
CD123	0
CD321	0
CD14	0
CD33	0
CD47	0
CD11c	0
CD7	0
CD15	0
CD16	0
CD44	0
CD38	0
CD13	0
CD3	0
CD61	0
CD117	0
CD49d	0
HLA-DR	0
CD64	0
CD41	0
Viability	0
file_number	0
event_number	0
label	161443
individual	0
dtype:	int64

NULL VS NOT NULL

```
In [ ]: df = pd.DataFrame(df)
null_values = df.isnull().sum()
non_null_values = df.notnull().sum()
plot_data = pd.DataFrame({
    'Null Values': null_values,
    'Non-Null Values': non_null_values
})
plot_data.plot(kind='bar', figsize=(12, 6))
plt.title('Null and Non-Null Values in Each Column')
```

```
plt.xlabel('Columns')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(loc='upper right')
plt.show()
```



Dropping Unnecessary Column

```
In [ ]: df = df.drop(columns=['Event', 'Time', 'individual', 'file_number', 'event_number'])
```

Class Label Distribution

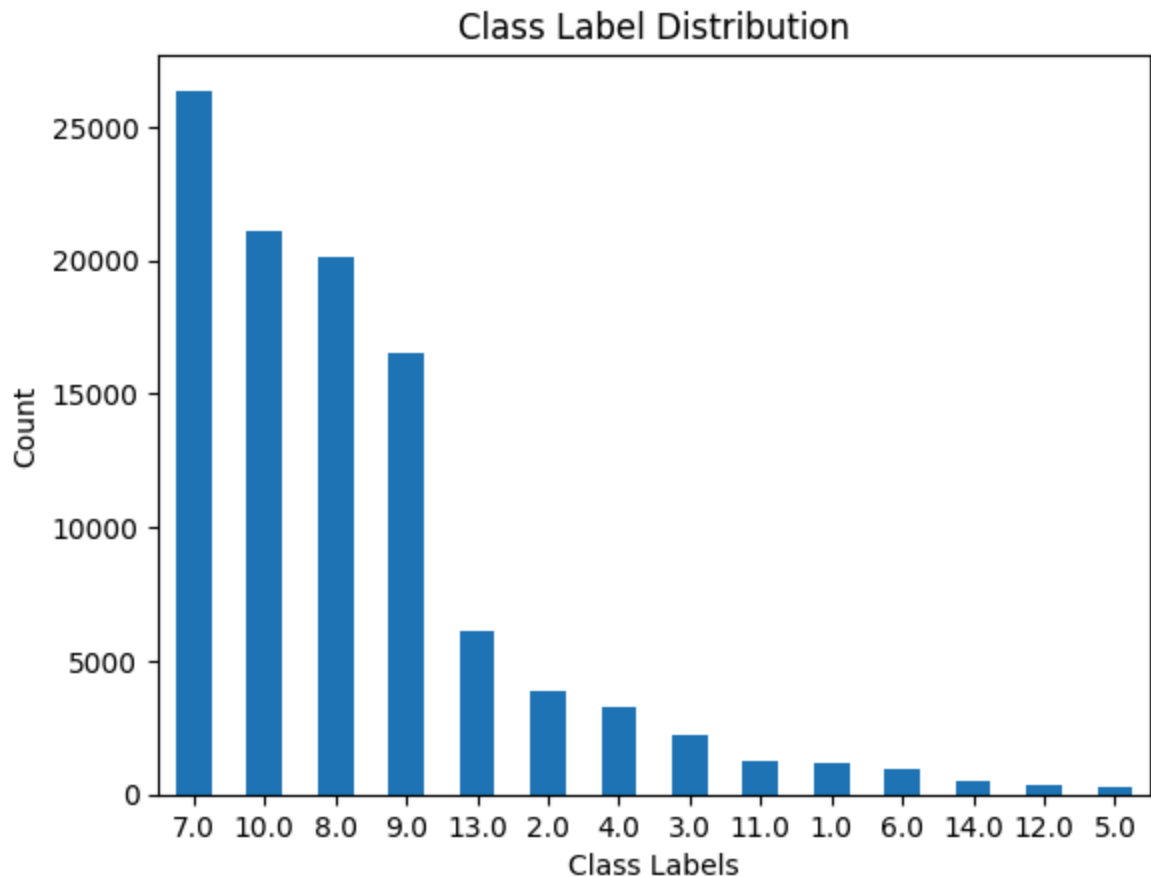
```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

class_counts = data['label'].value_counts()

class_counts.plot(kind='bar')

plt.title('Class Label Distribution')
plt.ylabel('Count')
plt.xlabel('Class Labels')
plt.xticks(rotation=0)
plt.show()
```



Histogram of each feature

```
In [ ]: import pandas as pd
import matplotlib.pyplot as plt

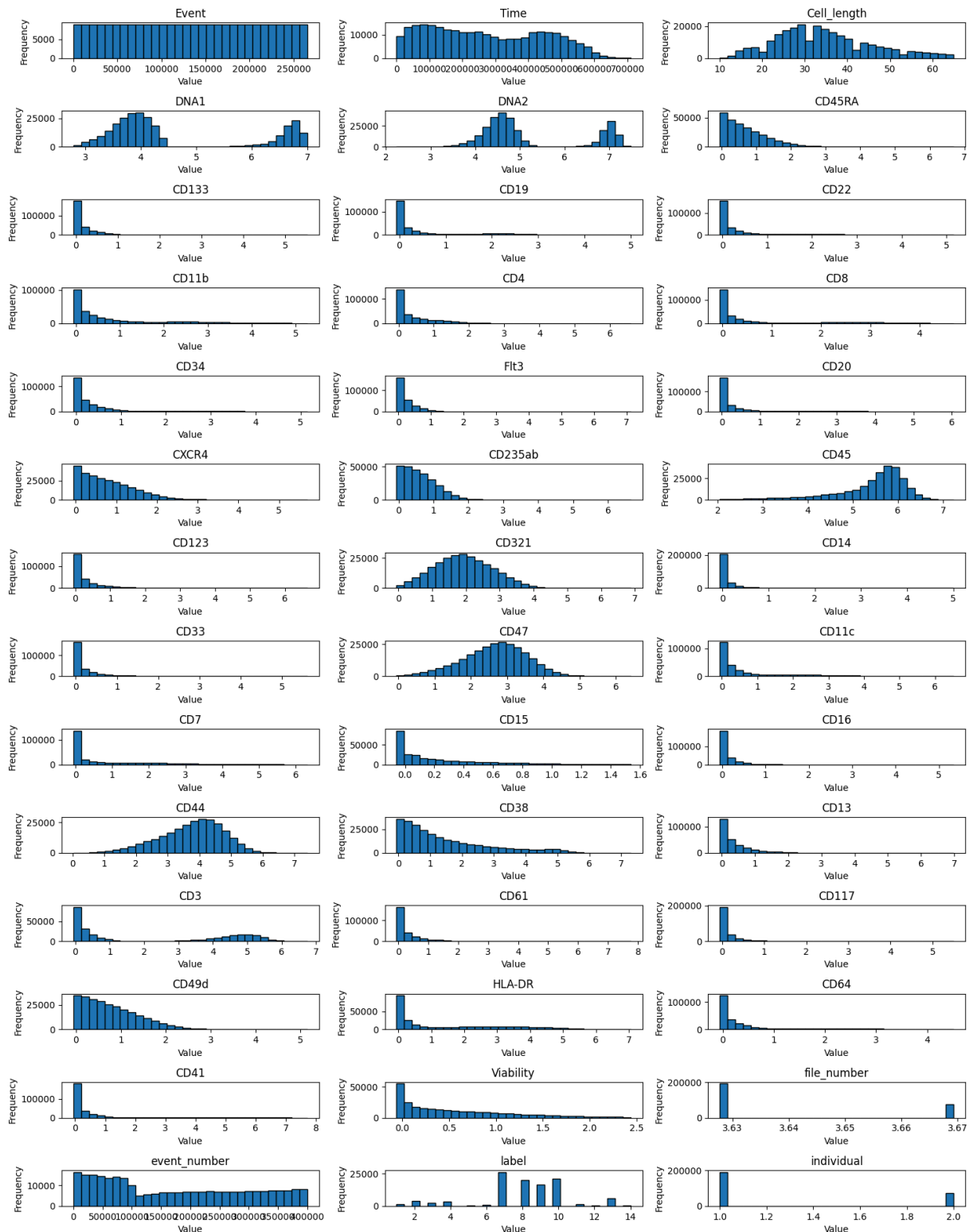
data = pd.read_csv('/content/drive/MyDrive/Datasets/Levine_32dim.fcs.csv')

# Select only numerical columns for histogram plotting
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns

# Set up the figure for subplots
plt.figure(figsize=(15, 20))

# Iterate through numerical columns and create a histogram for each
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(len(numerical_columns)//3 + 1, 3, i)
    plt.hist(data[column], bins=30, edgecolor='black')
    plt.title(column)
    plt.xlabel('Value')
    plt.ylabel('Frequency')

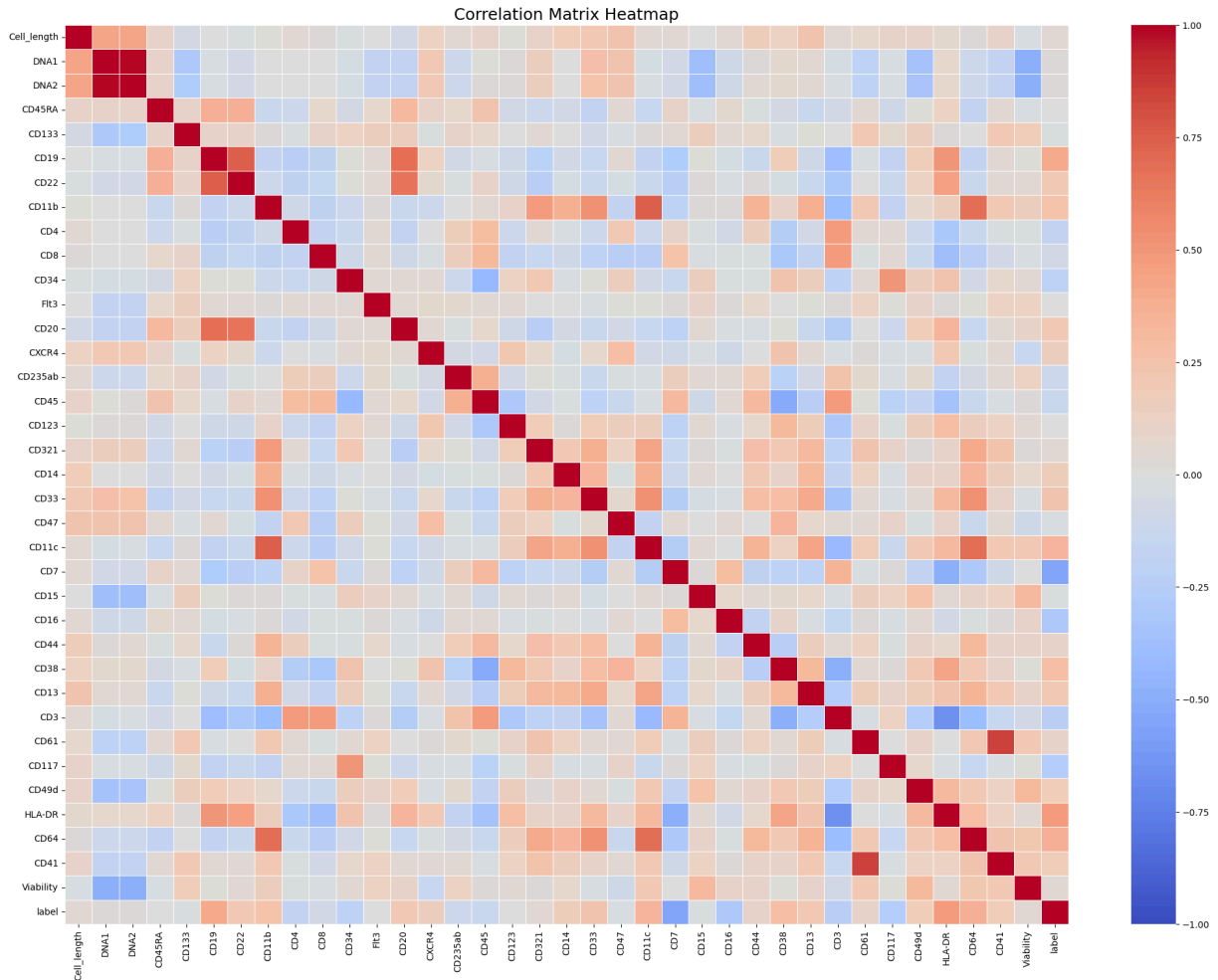
plt.tight_layout()
plt.show()
```

Correlation matrix

```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt
# Compute the correlation matrix
correlation_matrix = df.corr()
# Display the correlation matrix
correlation_matrix.round(2)
```

```
# Set the figure size
plt.figure(figsize=(25, 18))
# Create a heatmap of the correlation matrix
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', vmin=-1, vmax=1)
# Add a title
plt.title('Correlation Matrix Heatmap', fontsize=18)
# Show the plot
plt.show()
```



Setting the threshold

```
In [ ]: def correlation(dataset, threshold):
    col_corr = set() # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested
                colname = corr_matrix.columns[i] # getting the name of column
                col_corr.add(colname)
    return col_corr
```

```
In [ ]: corr_features = correlation(df, 0.8)
len(set(corr_features))
```

Out[]: 2

```
In [ ]: corr_features
```

Out[]: {'CD41', 'DNA2'}

Finding the range of each column

```
In [ ]: # Select only numerical columns
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Create a DataFrame to store max, min, and range values
summary_df = pd.DataFrame({
    'Max': df[numerical_columns].max(),
    'Min': df[numerical_columns].min(),
    'Range': df[numerical_columns].max() - df[numerical_columns].min()
})

# Display the summary DataFrame
print("Summary of Max, Min, and Range for Each Numerical Column:")
print(summary_df)
```

Summary of Max, Min, and Range for Each Numerical Column:

	Max	Min	Range
Cell_length	65.000000	10.000000	55.000000
DNA1	7.001489	2.786488	4.215001
DNA2	7.472308	2.236450	5.235858
CD45RA	6.691197	-0.057305	6.748502
CD133	5.527494	-0.058081	5.585575
CD19	4.990085	-0.058089	5.048174
CD22	5.160477	-0.057342	5.217819
CD11b	5.260789	-0.058236	5.319025
CD4	6.581762	-0.057751	6.639513
CD8	4.693694	-0.058003	4.751697
CD34	5.147996	-0.058008	5.206004
Flt3	7.117323	-0.057884	7.175207
CD20	6.051411	-0.058132	6.109543
CXCR4	5.696674	-0.057042	5.753717
CD235ab	6.646699	-0.057612	6.704311
CD45	7.238076	2.040243	5.197833
CD123	6.640626	-0.058003	6.698630
CD321	6.867388	-0.053552	6.920940
CD14	5.006121	-0.057954	5.064075
CD33	5.612469	-0.058079	5.670548
CD47	6.402488	-0.055087	6.457575
CD11c	6.520939	-0.058053	6.578992
CD7	6.319219	-0.058162	6.377381
CD15	1.534151	-0.058077	1.592227
CD16	5.338305	-0.057780	5.396085
CD44	7.404564	0.026061	7.378503
CD38	7.293085	-0.057194	7.350279
CD13	6.981187	-0.057728	7.038915
CD3	6.748362	-0.058241	6.806603
CD61	7.748498	-0.057642	7.806139
CD117	5.502125	-0.057668	5.559793
CD49d	5.153438	-0.058064	5.211502
HLA-DR	7.052507	-0.057974	7.110481
CD64	4.517843	-0.058199	4.576042
CD41	7.718288	-0.058244	7.776532
Viability	2.433031	-0.057979	2.491010
label	14.000000	1.000000	13.000000

Boxplot

```
In [ ]: n_cols = 6 # You can adjust this to 7, 8, 9, or 10 as needed
n_rows = (len(numerical_columns) + n_cols - 1) // n_cols # Calculate number

# Set a suitable figure size
plt.figure(figsize=(n_cols * 5, n_rows * 5)) # Adjust height and width base

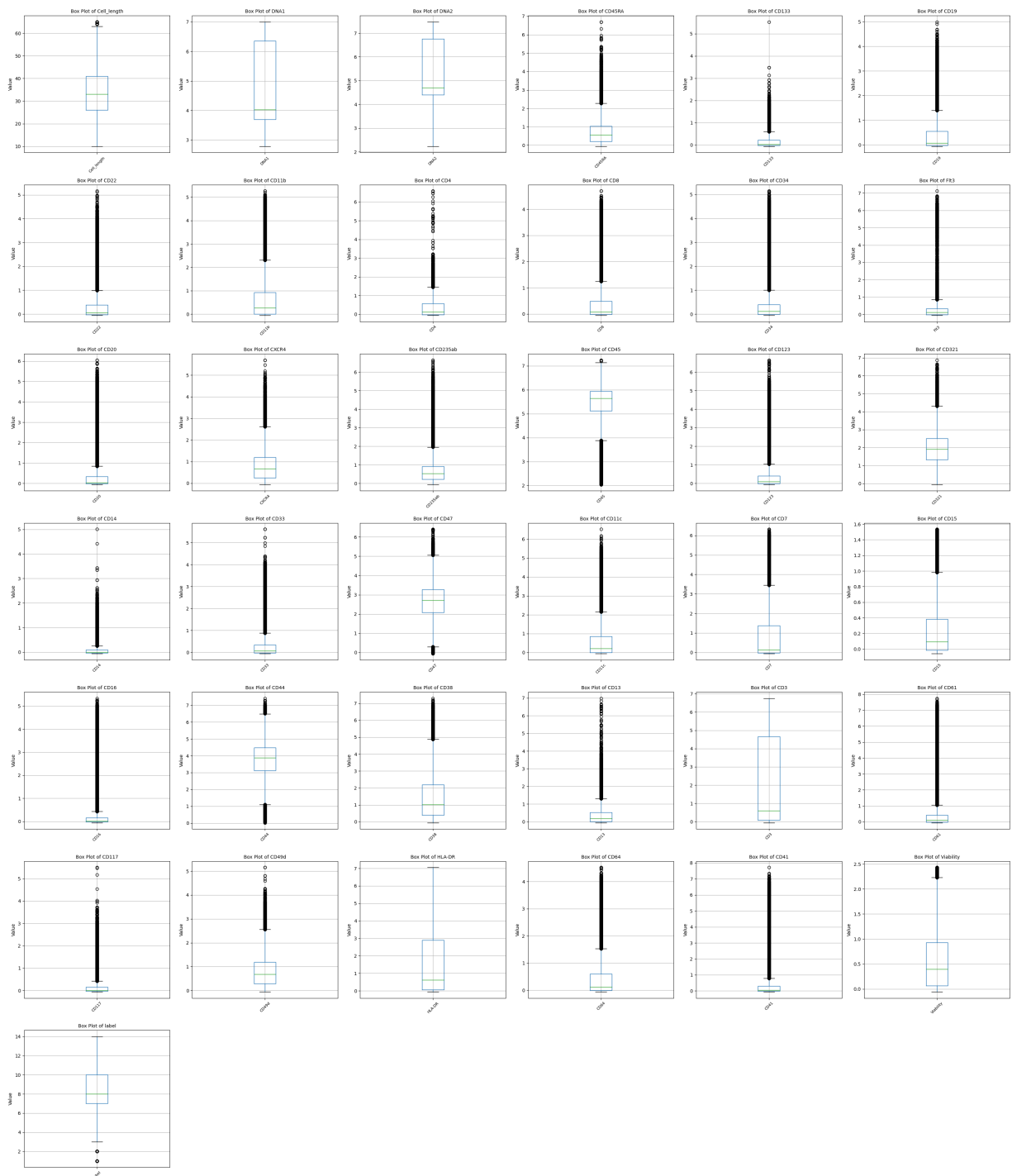
# Filter to include only valid numerical columns
valid_numerical_columns = [col for col in numerical_columns if col in df.col

# Loop through each valid numerical column to create individual box plots
for i, column in enumerate(valid_numerical_columns):
    plt.subplot(n_rows, n_cols, i + 1) # Create a grid of subplots
    df.boxplot(column=column)
```

```
plt.title(f'Box Plot of {column}', fontsize=10)
plt.ylabel('Value', fontsize=10)
plt.xticks(rotation=45, fontsize=8)
```

Adjust layout for better spacing

```
plt.tight_layout()
plt.show()
```



Skewness

```
In [ ]: import matplotlib.pyplot as plt
import seaborn as sns
```

```

from scipy.stats import skew
import pandas as pd

skewness = df.apply(skew)

# Function to categorize skewness
def categorize_skewness(value):
    if value > 0.5:
        return 'Right-skewed'
    elif value < -0.5:
        return 'Left-skewed'
    else:
        return 'Approximately symmetrical'

# Apply the categorization
skewness_category = skewness.apply(categorize_skewness)

# Display skewness and its categorization
skewness_df = pd.DataFrame({'Skewness': skewness, 'Category': skewness_category})
print(skewness_df)

# Number of numerical columns
num_cols = len(df.columns)

# Create a grid of 6 plots per row
cols_per_row = 6
rows = (num_cols + cols_per_row - 1) // cols_per_row # Calculate the number of rows

# Create subplots
fig, axes = plt.subplots(rows, cols_per_row, figsize=(20, rows * 4))
axes = axes.flatten() # Flatten to make it easier to iterate through

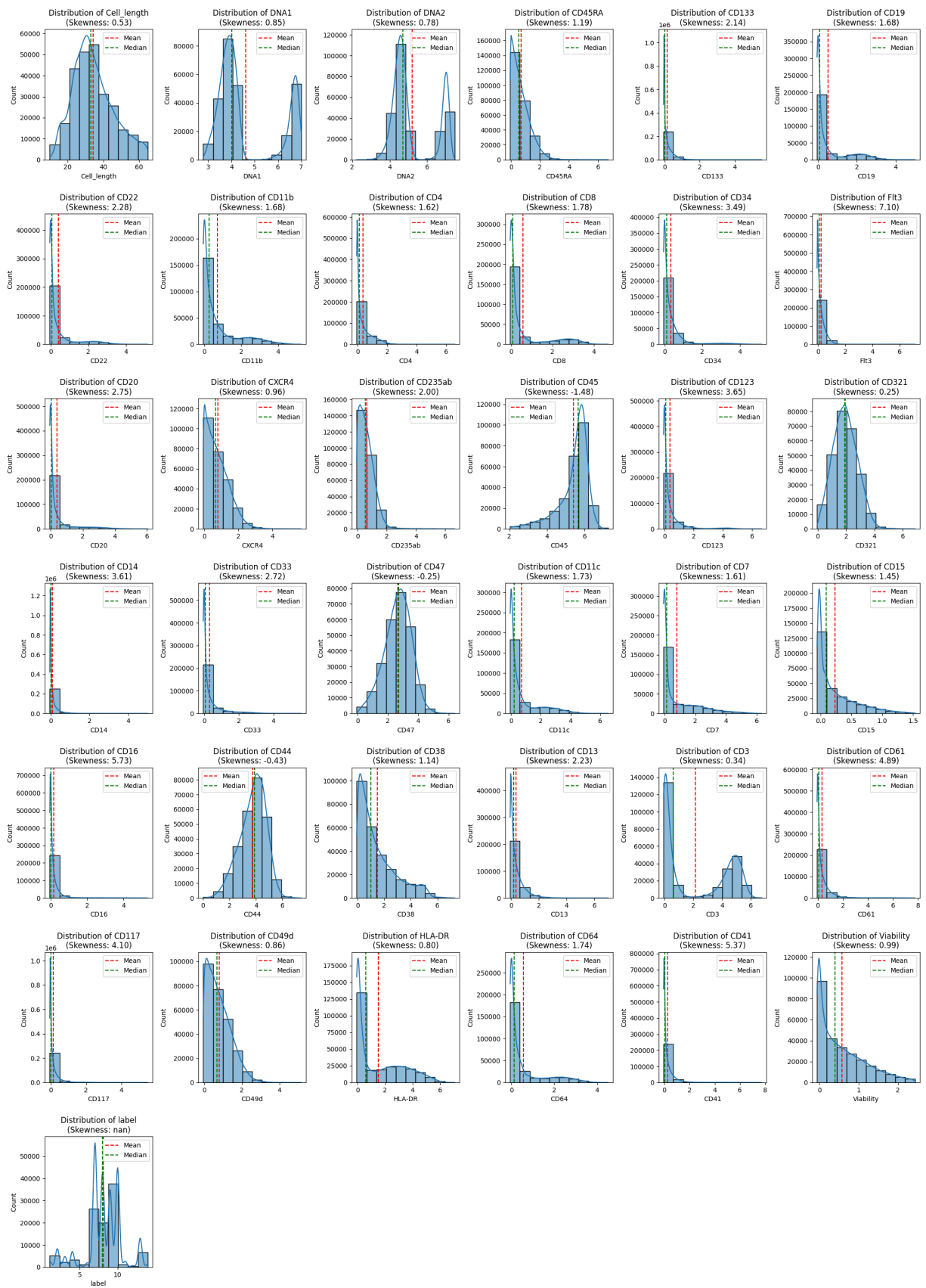
# Plot histograms for each numerical column
for i, col in enumerate(df.columns):
    sns.histplot(df[col], bins=10, kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}\n(Skewness: {skewness[col]:.2f})')
    axes[i].axvline(df[col].mean(), color='red', linestyle='--', label='Mean')
    axes[i].axvline(df[col].median(), color='green', linestyle='--', label='Median')
    axes[i].legend()

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
plt.tight_layout()
plt.show()

```

	Skewness	Category
Cell_length	0.527832	Right-skewed
DNA1	0.845010	Right-skewed
DNA2	0.779167	Right-skewed
CD45RA	1.191595	Right-skewed
CD133	2.141953	Right-skewed
CD19	1.682609	Right-skewed
CD22	2.283181	Right-skewed
CD11b	1.679089	Right-skewed
CD4	1.622044	Right-skewed
CD8	1.775713	Right-skewed
CD34	3.492437	Right-skewed
Flt3	7.098151	Right-skewed
CD20	2.754699	Right-skewed
CXCR4	0.955342	Right-skewed
CD235ab	2.001479	Right-skewed
CD45	-1.484824	Left-skewed
CD123	3.648890	Right-skewed
CD321	0.247097	Approximately symmetrical
CD14	3.609006	Right-skewed
CD33	2.724977	Right-skewed
CD47	-0.250323	Approximately symmetrical
CD11c	1.733888	Right-skewed
CD7	1.606528	Right-skewed
CD15	1.445147	Right-skewed
CD16	5.733203	Right-skewed
CD44	-0.431589	Approximately symmetrical
CD38	1.141482	Right-skewed
CD13	2.234311	Right-skewed
CD3	0.342239	Approximately symmetrical
CD61	4.894707	Right-skewed
CD117	4.097508	Right-skewed
CD49d	0.856805	Right-skewed
HLA-DR	0.795359	Right-skewed
CD64	1.743733	Right-skewed
CD41	5.366314	Right-skewed
Viability	0.985417	Right-skewed
label	NaN	Approximately symmetrical



Kurtosis


```

In [ ]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kurtosis

# Calculate kurtosis for each column
kurtosis_values = df.apply(kurtosis, fisher=False) # Pearson kurtosis (normal distribution = 3)

# Create a DataFrame with kurtosis values
kurtosis_df = pd.DataFrame({'Column': df.columns, 'Kurtosis': kurtosis_values})

# Categorize the kurtosis values (Leptokurtic, Mesokurtic, Platykurtic)
def categorize_kurtosis(value):
    if value > 3:
        return 'Leptokurtic (heavy tails)'
    elif value < 3:
        return 'Platykurtic (light tails)'
    else:
        return 'Mesokurtic (normal tails)'

kurtosis_df['Category'] = kurtosis_df['Kurtosis'].apply(categorize_kurtosis)

# Print the kurtosis values and their categories
print(kurtosis_df)

# Number of numerical columns
num_cols = len(df.columns)

# Create a grid of 6 plots per row
cols_per_row = 6
rows = (num_cols + cols_per_row - 1) // cols_per_row # Calculate the number of rows

# Create subplots
fig, axes = plt.subplots(rows, cols_per_row, figsize=(20, rows * 4))
axes = axes.flatten() # Flatten the axes array to make iteration easier

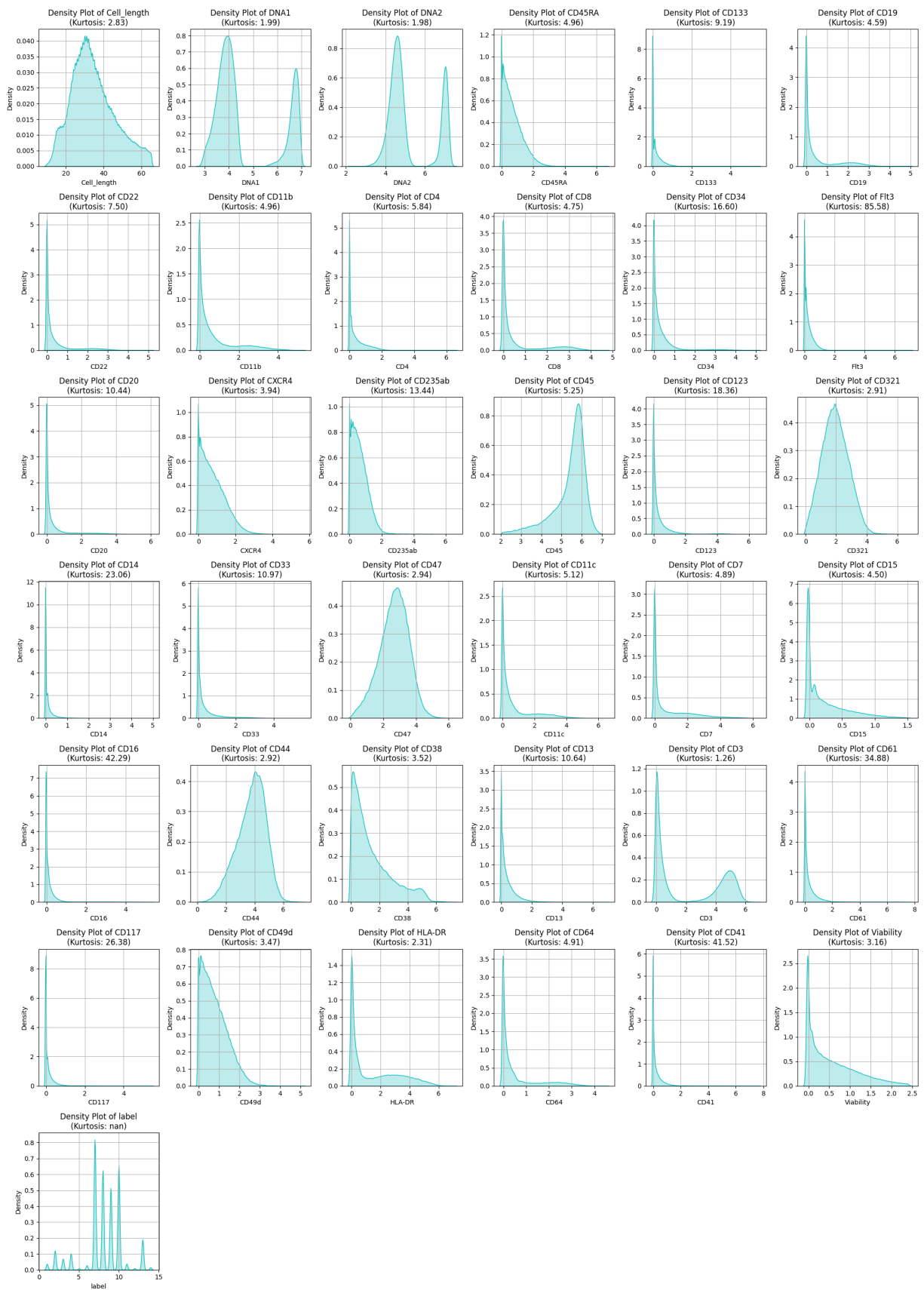
# Plot density for each column
for i, column in enumerate(df.columns):
    sns.kdeplot(df[column].dropna(), color='c', fill=True, bw_adjust=0.5, ax=axes[i])
    axes[i].set_title(f'Density Plot of {column}\n(Kurtosis: {kurtosis_df.loc[column, "Kurtosis"]})')
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Density')
    axes[i].grid(True)

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
plt.tight_layout()
plt.show()

```

	Column	Kurtosis	Category
Cell_length	Cell_length	2.834033	Platykurtic (light tails)
DNA1	DNA1	1.994037	Platykurtic (light tails)
DNA2	DNA2	1.975021	Platykurtic (light tails)
CD45RA	CD45RA	4.964272	Leptokurtic (heavy tails)
CD133	CD133	9.190066	Leptokurtic (heavy tails)
CD19	CD19	4.590887	Leptokurtic (heavy tails)
CD22	CD22	7.500223	Leptokurtic (heavy tails)
CD11b	CD11b	4.964495	Leptokurtic (heavy tails)
CD4	CD4	5.844261	Leptokurtic (heavy tails)
CD8	CD8	4.745776	Leptokurtic (heavy tails)
CD34	CD34	16.596416	Leptokurtic (heavy tails)
Flt3	Flt3	85.583534	Leptokurtic (heavy tails)
CD20	CD20	10.435449	Leptokurtic (heavy tails)
CXCR4	CXCR4	3.936307	Leptokurtic (heavy tails)
CD235ab	CD235ab	13.440586	Leptokurtic (heavy tails)
CD45	CD45	5.246770	Leptokurtic (heavy tails)
CD123	CD123	18.361217	Leptokurtic (heavy tails)
CD321	CD321	2.914593	Platykurtic (light tails)
CD14	CD14	23.062535	Leptokurtic (heavy tails)
CD33	CD33	10.967536	Leptokurtic (heavy tails)
CD47	CD47	2.943834	Platykurtic (light tails)
CD11c	CD11c	5.117156	Leptokurtic (heavy tails)
CD7	CD7	4.885115	Leptokurtic (heavy tails)
CD15	CD15	4.504387	Leptokurtic (heavy tails)
CD16	CD16	42.287749	Leptokurtic (heavy tails)
CD44	CD44	2.918792	Platykurtic (light tails)
CD38	CD38	3.521190	Leptokurtic (heavy tails)
CD13	CD13	10.637564	Leptokurtic (heavy tails)
CD3	CD3	1.264612	Platykurtic (light tails)
CD61	CD61	34.878020	Leptokurtic (heavy tails)
CD117	CD117	26.375108	Leptokurtic (heavy tails)
CD49d	CD49d	3.468119	Leptokurtic (heavy tails)
HLA-DR	HLA-DR	2.309924	Platykurtic (light tails)
CD64	CD64	4.910631	Leptokurtic (heavy tails)
CD41	CD41	41.521113	Leptokurtic (heavy tails)
Viability	Viability	3.156935	Leptokurtic (heavy tails)
label	label	NaN	Mesokurtic (normal tails)



Loading and Visualizing the MNIST Dataset

```
In [ ]: import tensorflow as tf

(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

print(f"Training images shape: {train_images.shape}")
print(f"Training labels shape: {train_labels.shape}")
print(f"Test images shape: {test_images.shape}")
print(f"Test labels shape: {test_labels.shape}")

import matplotlib.pyplot as plt

plt.imshow(train_images[0], cmap='Accent')
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>

11490434/11490434 ————— **0s** 0us/step

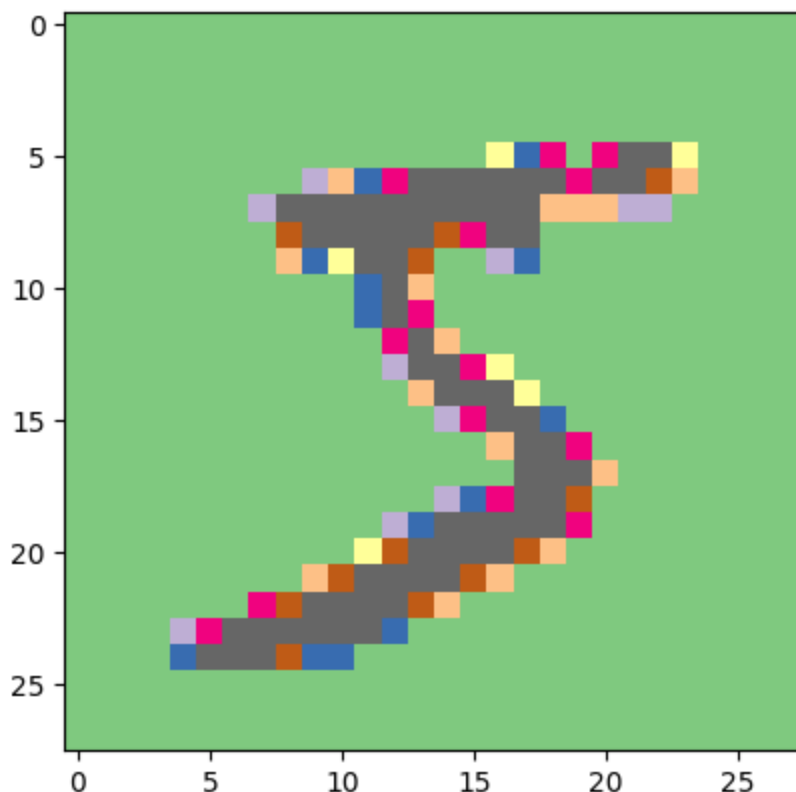
Training images shape: (60000, 28, 28)

Training labels shape: (60000,)

Test images shape: (10000, 28, 28)

Test labels shape: (10000,)

Out[]: <matplotlib.image.AxesImage at 0x7a9c10029540>



t-SNE Visualization of MNIST Dataset with Subset of Samples

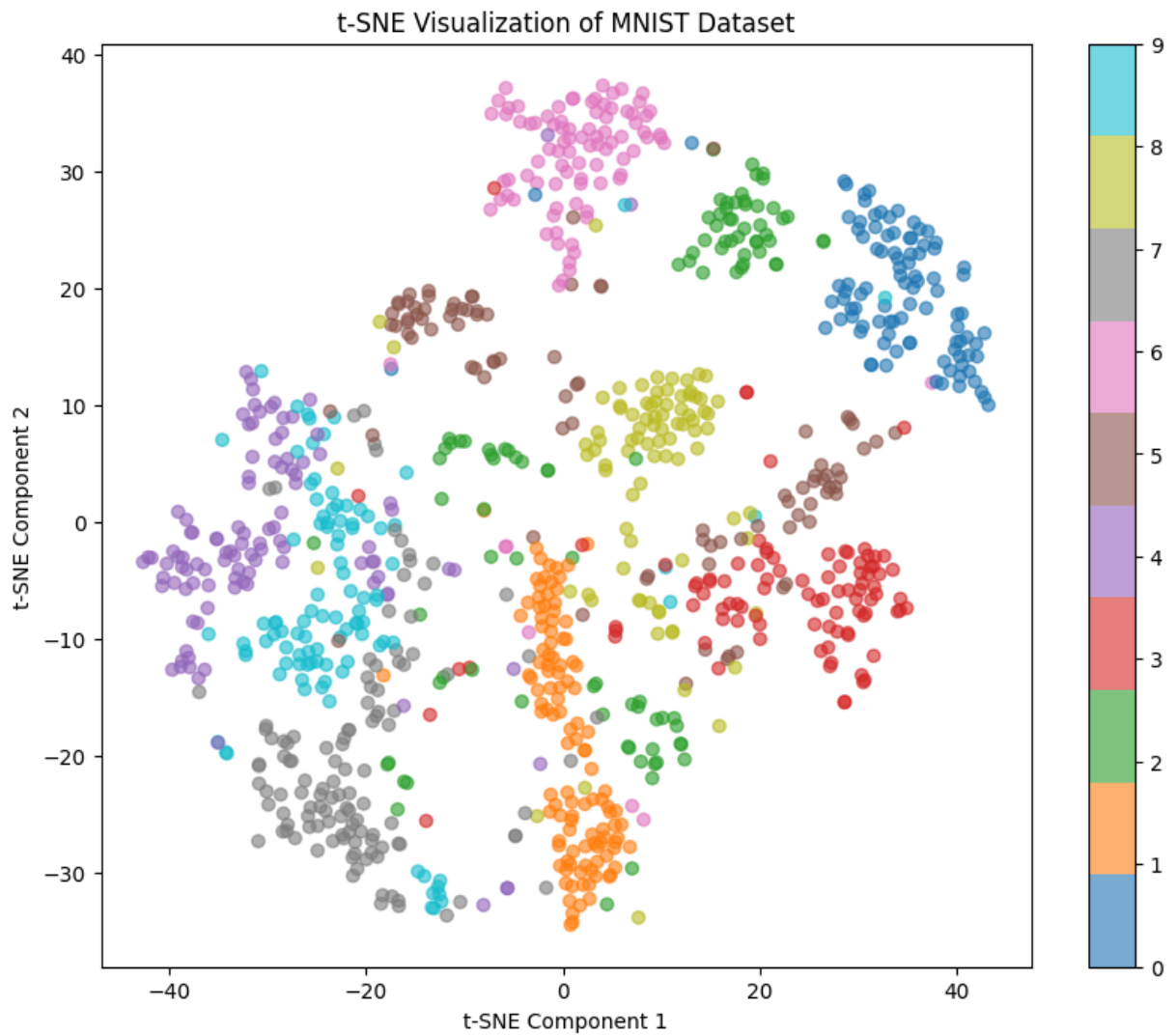
```
In [ ]: import tensorflow as tf
        from sklearn.manifold import TSNE
        import matplotlib.pyplot as plt
        import numpy as np

        (train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()
        train_images = train_images.astype('float32') / 255.0
        test_images = test_images.astype('float32') / 255.0

        n_samples = 1000
        train_images_flat = train_images[:n_samples].reshape(n_samples, -1)
        train_labels_subset = train_labels[:n_samples]

        tsne = TSNE(n_components=2, random_state=42, perplexity=30)
        train_images_tsne = tsne.fit_transform(train_images_flat)

        # plot the results
        plt.figure(figsize=(10, 8))
        scatter = plt.scatter(train_images_tsne[:, 0], train_images_tsne[:, 1], c=train_labels_subset)
        plt.colorbar(scatter, ticks=range(10))
        plt.title('t-SNE Visualization of MNIST Dataset')
        plt.xlabel('t-SNE Component 1')
        plt.ylabel('t-SNE Component 2')
        plt.show()
```



Standardizing the data

```
In [ ]: import pandas as pd
from sklearn.preprocessing import StandardScaler
# Initialize the StandardScaler
scaler = StandardScaler()

scaled_data = scaler.fit_transform(df)

scaled_df = pd.DataFrame(scaled_data, columns=df.columns)

# Display the first few rows of the standardized data
print(scaled_df.head())
```

	Cell_length	DNA1	DNA2	CD45RA	CD133	CD19	CD22
\							
0	-1.087702	-0.164453	-0.505101	-0.862639	-0.677085	-0.601774	-0.434227
1	0.047999	-0.202977	-0.331737	0.021706	-0.710621	-0.613387	-0.423702
2	-0.214086	-0.585171	-0.705816	-0.138826	-0.687231	-0.507832	-0.577727
3	-0.476171	-0.267476	-0.320127	-0.417630	-0.669470	-0.614562	-0.579163
4	-0.825617	-0.479916	-0.601444	-1.144201	-0.679832	-0.500173	0.129202

	CD11b	CD4	CD8	...	CD13	CD3	CD61	CD117
\								
0	-0.711371	-0.007722	-0.044861	...	-0.665941	-0.968994	-0.479732	-0.249511
1	0.096608	-0.778973	-0.573653	...	2.197090	-0.973786	1.262168	-0.132623
2	-0.704150	-0.727475	-0.568752	...	-0.312734	-0.813227	-0.120582	-0.271310
3	0.023115	-0.747355	-0.507239	...	0.294199	-0.927985	-0.532490	-0.206663
4	0.392818	0.357861	-0.531946	...	0.438207	-0.894158	-0.242836	-0.438757

	CD49d	HLA-DR	CD64	CD41	Viability	label
0	0.093316	0.084209	-0.626606	-0.427371	0.132927	-2.895698
1	-0.951407	-0.608084	-0.457614	0.982491	-0.014673	-2.895698
2	2.854812	-0.126003	-0.632890	-0.441068	0.124292	-2.895698
3	0.866341	-0.815301	-0.635690	-0.466392	-1.011569	-2.895698
4	-0.978326	-0.781769	-0.534854	-0.489807	-0.486233	-2.895698

[5 rows x 37 columns]

PCA for 2D

```
In [ ]: import pandas as pd
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Scale the data using StandardScaler
# Use all columns except the 'label' column
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop(columns=['label']))

# Apply PCA to reduce dimensions to 2 components
pca = PCA(n_components=2)
pca_results = pca.fit_transform(scaled_data)

# Add PCA results back to the original DataFrame
df['PCA1'] = pca_results[:, 0]
df['PCA2'] = pca_results[:, 1]

# Create a scatter plot of the PCA results
plt.figure(figsize=(10, 8))
scatter = plt.scatter(
    df['PCA1'], df['PCA2'],
    c=df['label'], # Use the label column for coloring
    cmap='viridis', # 'viridis' colormap for smooth gradients
    alpha=0.6 # Set transparency for better visualization
)
```

```

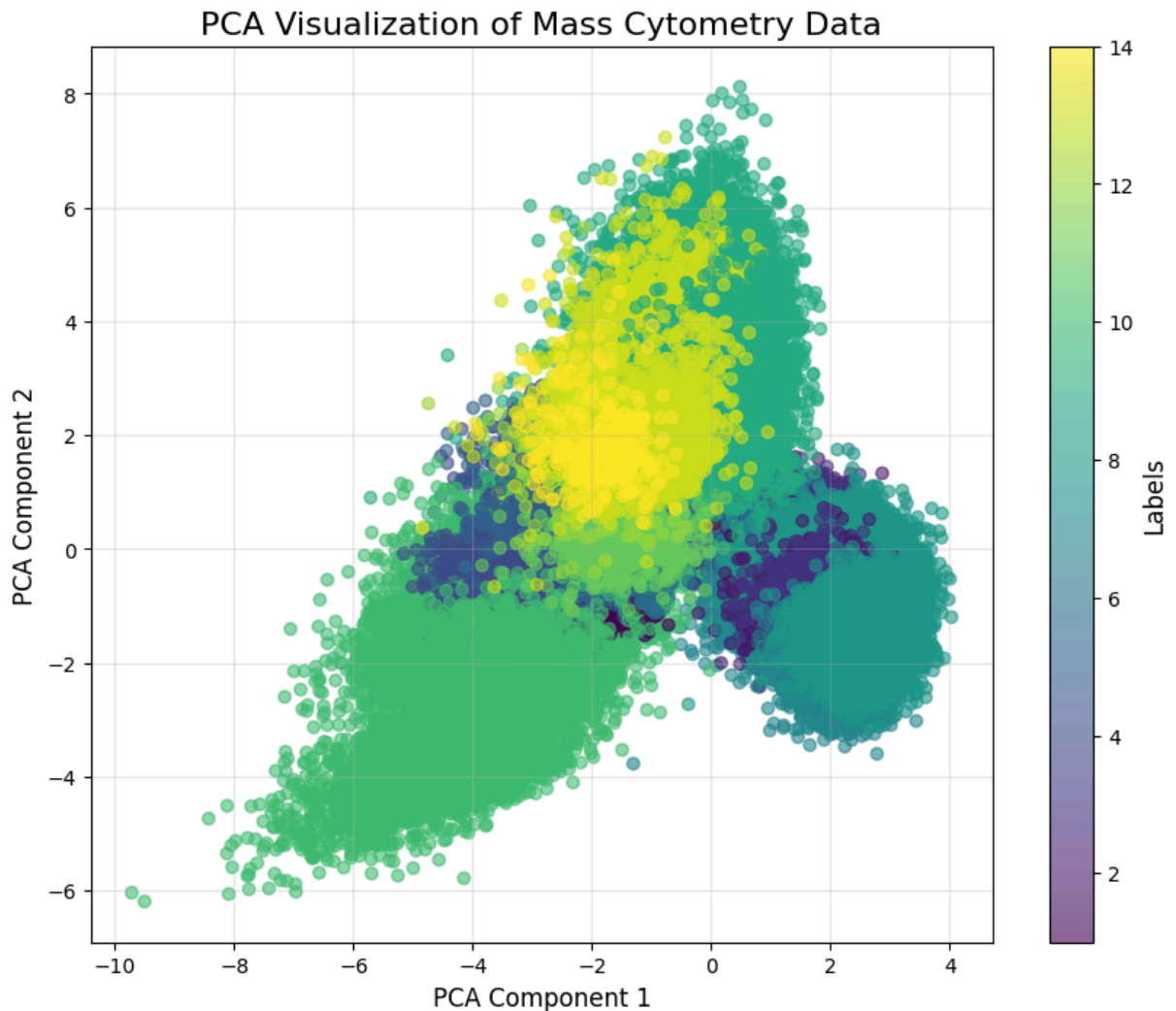
# Add a colorbar to explain the labels
cbar = plt.colorbar(scatter)
cbar.set_label('Labels', fontsize=12)

# Add plot title and axis labels
plt.title('PCA Visualization of Mass Cytometry Data', fontsize=16)
plt.xlabel('PCA Component 1', fontsize=12)
plt.ylabel('PCA Component 2', fontsize=12)

# Add grid for clarity
plt.grid(alpha=0.3)

# Show the final plot
plt.show()

```



PCA for 3D

```

In [ ]: import pandas as pd
import numpy as np # Ensure this is imported for mathematical operations
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # For 3D plotting
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

```



```

# Scale the data using StandardScaler
# Use all columns except 'label' for scaling
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df.drop(columns=['label']))

# Apply PCA with 4 components
pca = PCA(n_components=4)
pca_results = pca.fit_transform(scaled_data)

# Add PCA results back to the DataFrame
df['PCA1'] = pca_results[:, 0]
df['PCA2'] = pca_results[:, 1]
df['PCA3'] = pca_results[:, 2]
df['PCA4'] = pca_results[:, 3]

# Print PCA explained variance results
explained_variance = pca.explained_variance_ratio_
cumulative_variance = explained_variance.cumsum()
standard_deviation = pca.singular_values_ / np.sqrt(len(df) - 1)

print(f"Standard deviation: {standard_deviation}")
print(f"Proportion of Variance: {explained_variance}")
print(f"Cumulative Proportion: {cumulative_variance}")

# 3D scatter plot of the PCA results (PC1, PC2, PC3)
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot using the 'viridis' colormap
scatter = ax.scatter(
    df['PCA1'], df['PCA2'], df['PCA3'],
    c=df['label'], cmap='viridis', alpha=0.6
)

# Add title and axis labels
ax.set_title('PCA Visualization of Mass Cytometry Data (3D: PC1, PC2, PC3)',
ax.set_xlabel('PCA Component 1', fontsize=12)
ax.set_ylabel('PCA Component 2', fontsize=12)
ax.set_zlabel('PCA Component 3', fontsize=12)

# Add a colorbar with a label
cbar = fig.colorbar(scatter, ax=ax, label='Labels')

# Show the plot
plt.show()

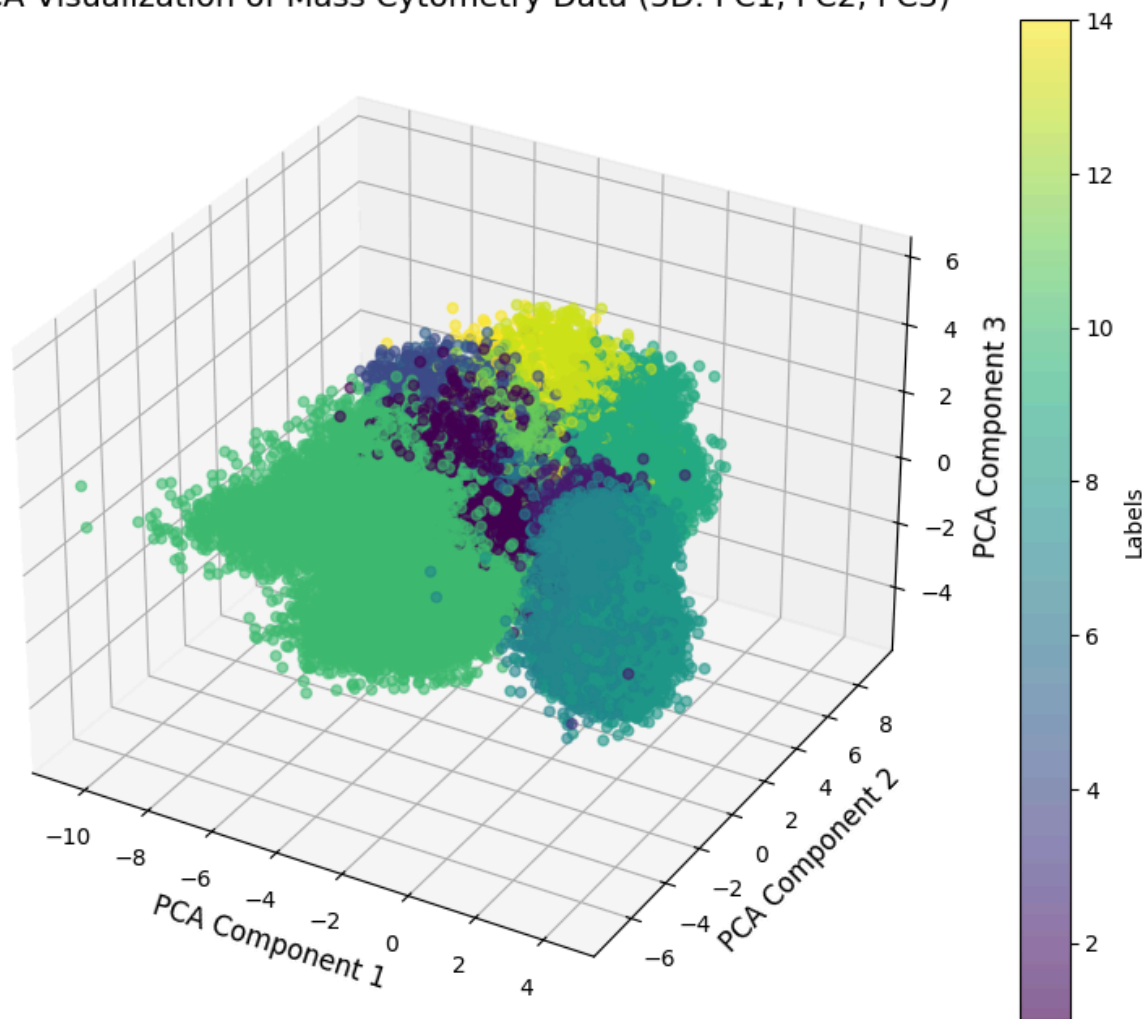
```

```

Standard deviation: [2.53583711 2.20245521 1.89526477 1.61107568]
Proportion of Variance: [0.16922225 0.12765239 0.09452671 0.06830408]
Cumulative Proportion: [0.16922225 0.29687464 0.39140135 0.45970543]

```

PCA Visualization of Mass Cytometry Data (3D: PC1, PC2, PC3)



T-SNE

```
In [ ]: pip install openTSNE
```

Collecting openTSNE

Downloading openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.8 kB)

Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.26.4)

Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.5.2)

Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.13.1)

Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->openTSNE) (1.4.2)

Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->openTSNE) (3.5.0)

Downloading openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)

3.0/3.0 MB 22.8 MB/s eta 0:00:00

Installing collected packages: openTSNE

Successfully installed openTSNE-1.0.2

```

In [ ]: from openTSNE import TSNE
import matplotlib.pyplot as plt

# Step 1: Reduce dimensionality with PCA
from sklearn.decomposition import PCA

pca = PCA(n_components=20)
pca_result = pca.fit_transform(scaled_data)

# Step 2: Apply t-SNE using openTSNE (multithreaded)
# `n_jobs=-1` uses all available CPU cores
tsne = TSNE(
    n_components=2, perplexity=30, n_jobs=-1, random_state=42, initialization='random'
)
tsne_results = tsne.fit(pca_result)

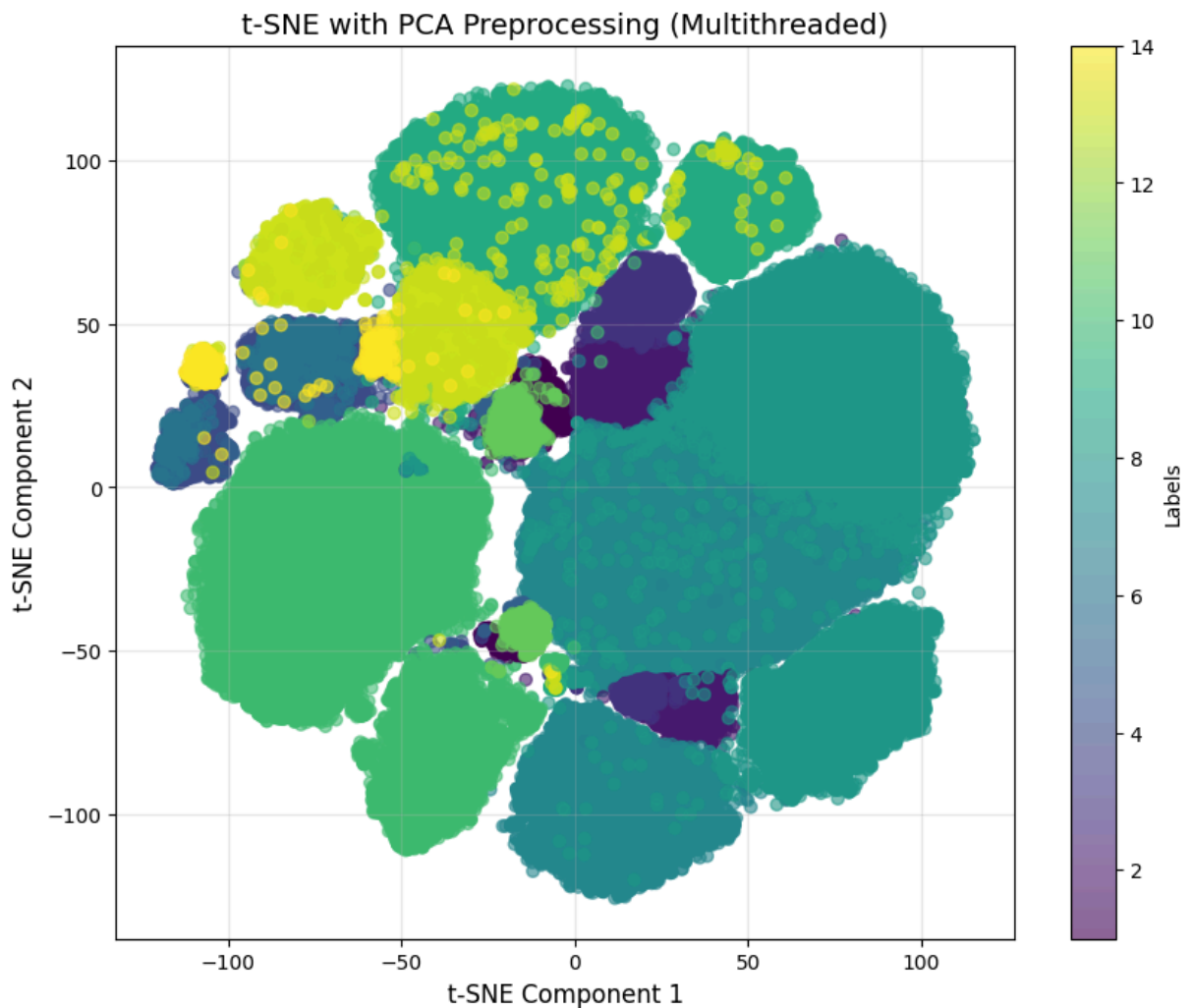
# Step 3: Add t-SNE results to the dataframe
df['TSNE1'] = tsne_results[:, 0]
df['TSNE2'] = tsne_results[:, 1]

# Step 4: Plot the t-SNE results
plt.figure(figsize=(10, 8))
scatter = plt.scatter(
    df['TSNE1'], df['TSNE2'],
    c=df['label'], cmap='viridis', alpha=0.6
)

# Add a colorbar and labels
cbar = plt.colorbar(scatter, label='Labels')
plt.title('t-SNE with PCA Preprocessing (Multithreaded)', fontsize=14)
plt.xlabel('t-SNE Component 1', fontsize=12)
plt.ylabel('t-SNE Component 2', fontsize=12)
plt.grid(alpha=0.3)

# Show the plot
plt.show()

```



Applying Binary Mask, Shuffled Output and Corrupted DataFrame on example data

```
In [ ]: import pandas as pd
import numpy as np

np.random.seed(42)
demo_data = pd.DataFrame({
    'A': [5, 11, 18, 8],
    'B': [10, 40, 15, 30],
    'C': [9, 25, 35, 20]
})

p_m = 0.5

data_array = demo_data.values
mask = np.random.binomial(1, p_m, data_array.shape)
print("Generated Mask (1 represents masked values):\n", mask)

masked_data = np.where(mask == 1, np.nan, data_array)
masked_demo_data = pd.DataFrame(masked_data, columns=demo_data.columns)
```

```
print("\nOriginal DataFrame:\n", demo_data)
print("\nMasked DataFrame:\n", masked_demo_data)
```

Generated Mask (1 represents masked values):

```
[[0 1 1]
 [1 0 0]
 [0 1 1]
 [1 0 1]]
```

Original DataFrame:

	A	B	C
0	5	10	9
1	11	40	25
2	18	15	35
3	8	30	20

Masked DataFrame:

	A	B	C
0	5.0	NaN	NaN
1	NaN	40.0	25.0
2	18.0	NaN	NaN
3	NaN	30.0	NaN

```
In [ ]: import pandas as pd
import numpy as np

data = {
    'A': [1, 2, 3, 4, 5],
    'B': [10, 20, 30, 40, 50],
    'C': [100, 200, 300, 400, 500],
    'D': [120, 300, 231, 450, 200],
    'E': [12, 30, 31, 40, 20]
}
df = pd.DataFrame(data)

shuffled_df = df.apply(np.random.permutation)

print("Original DataFrame:")
print(df)
print("\nDataFrame with shuffled column values:")
print(shuffled_df)
```

Original DataFrame:

	A	B	C	D	E
0	1	10	100	120	12
1	2	20	200	300	30
2	3	30	300	231	31
3	4	40	400	450	40
4	5	50	500	200	20

DataFrame with shuffled column values:

	A	B	C	D	E
0	1	30	300	300	12
1	3	20	400	450	20
2	5	50	100	231	31
3	2	10	500	200	40
4	4	40	200	120	30

```
In [ ]: import pandas as pd
import numpy as np

data = {
    'A': [1, 2, 3, 4, 5],
    'B': [10, 20, 30, 40, 50],
    'C': [100, 200, 300, 400, 500],
    'D': [1000, 2000, 3000, 4000, 5000],
    'E': [10000, 20000, 30000, 40000, 50000]
}
x = pd.DataFrame(data)

m = pd.DataFrame(np.random.binomial(1, 0.5, x.shape), columns=x.columns)
x_shuffled = x.apply(np.random.permutation)
x_corrupted = x * (1 - m) + x_shuffled * m

print("Original DataFrame (x):")
print(x)
print("\nBinary Mask (m):")
print(m)
print("\nShuffled DataFrame (x_shuffled):")
print(x_shuffled)
print("\nCorrupted DataFrame (x_corrupted):")
print(x_corrupted)
```

Original DataFrame (x):

	A	B	C	D	E
0	1	10	100	1000	10000
1	2	20	200	2000	20000
2	3	30	300	3000	30000
3	4	40	400	4000	40000
4	5	50	500	5000	50000

Binary Mask (m):

	A	B	C	D	E
0	0	1	0	1	0
1	1	1	0	0	1
2	1	0	0	0	0
3	1	1	1	0	0
4	0	1	0	0	1

Shuffled DataFrame (x_shuffled):

	A	B	C	D	E
0	1	10	100	5000	30000
1	4	30	500	3000	20000
2	2	50	300	2000	10000
3	3	20	200	1000	40000
4	5	40	400	4000	50000

Corrupted DataFrame (x_corrupted):

	A	B	C	D	E
0	1	10	100	5000	10000
1	4	30	200	2000	20000
2	2	30	300	3000	30000
3	3	20	200	4000	40000
4	5	40	500	5000	50000

Applying Binary Mask, Shuffled Output and Corrupted DataFrame on Original Data

```
In [ ]: # Separate labeled and unlabeled data based on non-NaN and NaN values in the
df_labeled = df[df['label'].notnull()]
df_unlabeled = df[df['label'].isnull()]

# Print the shapes of labeled and unlabeled data
print("Labeled Data Shape:", df_labeled.shape)
print("Unlabeled Data Shape:", df_unlabeled.shape)
```

Labeled Data Shape: (104184, 37)

Unlabeled Data Shape: (161443, 37)

Split labelled dataset into x_test,x_train and y_test and y_train . train = 70% and test = 30%

```
In [ ]: from sklearn.model_selection import train_test_split
# Separate labeled and unlabeled data
```

```
df_labeled = df[df['label'].notnull()] # Labeled data
df_unlabeled = df[df['label'].isnull()] # Unlabeled data

# Separate features and target for labeled data
x_labeled = df_labeled.drop(columns=['label']) # Features
y_labeled = df_labeled['label'] # Target

# Separate features for unlabeled data
x_unlabeled = df_unlabeled.drop(columns=['label']) # Features (no labels)

# Split the labeled data into training and testing sets (e.g., 70% train, 30% test)
x_train, x_test, y_train, y_test = train_test_split(x_labeled, y_labeled, test_size=0.3, random_state=42)

print("\nTraining Features (x_train):\n", x_train.head())
print("\nTraining Labels (y_train):\n", y_train.head())
print("\nTesting Features (x_test):\n", x_test.head())
print("\nTesting Labels (y_test):\n", y_test.head())
```


Training Features (x_train):

	Cell_length	DNA1	DNA2	CD45RA	CD133	CD19	\
64113	25	3.899656	4.594272	0.976652	0.302811	0.154761	
82744	31	6.592998	6.901888	0.431481	-0.052898	-0.037690	
24294	41	3.543583	4.467671	0.377192	0.219081	0.245478	
7820	38	4.305227	4.881685	0.199351	0.100678	-0.025812	
43295	26	4.159271	4.861015	0.831285	0.191518	2.002712	

	CD22	CD11b	CD4	CD8	...	CD38	CD13	\
64113	-0.011676	3.180236	1.465950	0.086209	...	1.563844	0.480488	
82744	-0.029715	-0.040846	0.914311	0.022305	...	1.232765	0.100678	
24294	0.193328	0.075123	0.936352	-0.044813	...	0.486930	0.046766	
7820	-0.002898	1.437247	-0.013400	-0.001012	...	1.250272	0.731957	
43295	3.387782	0.179219	0.115231	-0.010963	...	2.883403	0.345273	

	CD3	CD61	CD117	CD49d	HLA-DR	CD64	CD41
64113	0.017010	0.051464	-0.003680	1.260410	0.700093	2.355886	0.125409
82744	5.722406	-0.036430	0.021689	0.034946	-0.055651	-0.023248	-0.054842
24294	4.061728	1.003383	0.406137	1.928676	-0.046849	0.229309	0.937020
7820	0.245939	-0.007282	1.421540	1.443145	2.461705	0.528679	0.072205
43295	0.226596	-0.040754	0.060944	1.294561	3.085858	-0.014128	0.479256

Viability

64113	0.840205
82744	-0.009329
24294	1.231347
7820	0.892480
43295	2.269233

[5 rows x 36 columns]

Training Labels (y_train):

64113	10.0
82744	7.0
24294	7.0
7820	6.0
43295	9.0

Name: label, dtype: float64

Testing Features (x_test):

	Cell_length	DNA1	DNA2	CD45RA	CD133	CD19	\
60544	49	3.618797	4.144135	0.198186	0.000282	0.253703	
50673	27	3.660988	4.497041	1.272625	0.129642	3.054480	
50682	23	3.854865	4.663734	1.527763	0.151383	2.361353	
1761	17	3.716473	4.465312	0.375236	-0.037150	-0.035385	
98760	32	6.826030	7.007709	0.223441	-0.048813	-0.018816	

	CD22	CD11b	CD4	CD8	...	CD38	CD13	\
60544	-0.018972	2.665005	0.079150	-0.002045	...	2.479135	1.419488	
50673	2.493220	0.189975	-0.024412	0.186744	...	2.212054	-0.020246	
50682	2.281009	0.528589	-0.014516	-0.002732	...	0.787080	-0.010742	
1761	0.127904	0.415204	0.226788	2.802413	...	0.042091	-0.018271	
98760	-0.045954	4.067125	0.004401	-0.012083	...	1.382377	0.154702	

	CD3	CD61	CD117	CD49d	HLA-DR	CD64	CD41
--	-----	------	-------	-------	--------	------	------

```

\
60544  0.643676  0.307357  0.208639  2.039954  2.847283  2.798986  1.090235
50673  0.054290  0.084448  0.033192  0.004637  4.488360  0.866820 -0.002174
50682  0.068448 -0.041903 -0.026017  0.109363  2.328828 -0.008223 -0.018680
1761   -0.039628 -0.001024 -0.017034  0.023385  0.120367  0.472159 -0.014919
98760  0.250393 -0.029816 -0.046020  0.140410  0.735830  1.011186 -0.044875

```

```

      Viability
60544  1.005784
50673  0.917810
50682  1.091297
1761   0.620643
98760  0.149759

```

[5 rows x 36 columns]

Testing Labels (y_test):

```

60544    10.0
50673     9.0
50682     9.0
1761      2.0
98760    10.0
Name: label, dtype: float64

```

```

In [ ]: from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit and transform the unlabeled data
x_unlabeled_scaled = scaler.fit_transform(x_unlabeled)

# Convert back to a DataFrame if needed (optional, for better readability)
x_unlabeled_scaled = pd.DataFrame(x_unlabeled_scaled, columns=x_unlabeled.co

```

```

In [ ]: from sklearn.model_selection import train_test_split
df_labeled = df[df['label'].notnull()] # Labeled data
df_unlabeled = df[df['label'].isnull()] # Unlabeled data

# Separate features and target for labeled data
X_labeled = df_labeled.drop(columns=['label']) # Features
y_labeled = df_labeled['label']               # Target

# Split the labeled data into training and testing sets (e.g., 70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X_labeled, y_labeled, te

# Print the shapes of the training and testing sets
print("Shape of Training Features (X_train):", X_train.shape)
print("Shape of Training Labels (y_train):", y_train.shape)
print("Shape of Testing Features (X_test):", X_test.shape)

```

```

Shape of Training Features (X_train): (72928, 36)
Shape of Training Labels (y_train): (72928,)
Shape of Testing Features (X_test): (31256, 36)

```

Logistic Regression Model

```
In [ ]: from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import log_loss

        def logit(x_train, y_train, x_test):
            """Logistic Regression.

            Args:
                x_train: Training features.
                y_train: Training labels.
                x_test: Testing features.

            Returns:
                y_test_hat: Predicted probabilities for x_test.
            """
            # Convert labels into proper format
            if len(y_train.shape) > 1:
                y_train = donvert_matrix_to_vector(y_train)

            # Define and fit the model on the training dataset
            model = LogisticRegression()
            model.fit(x_train, y_train)

            # Predict probabilities on x_test
            y_test_hat = model.predict_proba(x_test)

            return y_test_hat
```

```
In [ ]: y_test_prob = logit(X_train, y_train, X_test)

        # Display the probabilities
        print("Predicted probabilities for the test set:")
        print(y_test_prob)

        # Compute log loss
        log_loss_value = log_loss(y_test, y_test_prob)

        # Display log loss
        print("Log loss for the test set:", log_loss_value)
```

```

Predicted probabilities for the test set:
[[1.27832255e-12 2.06977665e-16 3.99046638e-17 ... 7.92486068e-13
 3.66276613e-14 1.83700781e-13]
 [3.53829724e-14 5.62561775e-14 9.40919132e-16 ... 1.07032765e-11
 2.42897888e-04 1.52985856e-10]
 [9.66721886e-11 2.33132685e-10 3.73727689e-12 ... 1.13644612e-10
 1.28665515e-06 3.89190497e-11]
 ...
 [8.97193682e-08 1.11777043e-05 1.13462283e-08 ... 2.18504192e-08
 2.32788580e-10 2.44061608e-10]
 [4.43450554e-09 3.80180801e-10 2.59394355e-11 ... 1.75572500e-08
 5.96201221e-06 2.02390897e-07]
 [2.07984818e-09 7.62212185e-09 7.21995065e-11 ... 3.49653489e-11
 5.64646108e-08 4.47350113e-12]]
Log loss for the test set: 0.033144266653965554

```

```

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:46
9: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

```

```

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(

```

XGBoost Model

```

In [ ]: from xgboost import XGBClassifier
from sklearn.metrics import log_loss
import numpy as np

def xgboost_model(x_train, y_train, x_test):
    """XGBoost Classifier.

    Args:
        x_train: Training features.
        y_train: Training labels.
        x_test: Testing features.

    Returns:
        y_test_prob: Predicted probabilities for x_test.
    """
    # Convert labels to proper format and zero-based index if necessary
    if len(y_train.shape) > 1:
        y_train = donvert_matrix_to_vector(y_train)

    y_train = y_train.astype(int) - 1 # Convert to integer and zero-based i

    # Define and fit the XGBoost model on the training dataset
    model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
    model.fit(x_train, y_train)

    # Predict probabilities on x_test

```

```

y_test_prob = model.predict_proba(x_test)

return y_test_prob

# Example usage
# Assuming y_test is the true labels for X_test
y_test_zero_based = y_test.astype(int) - 1 # Adjust y_test for log loss cal
y_test_prob = xgboost_model(X_train, y_train, X_test)

# Display the probabilities
print("Predicted probabilities for the test set:")
print(y_test_prob)

# Compute log loss
log_loss_value = log_loss(y_test_zero_based, y_test_prob)
print("Log loss for the test set:", log_loss_value)

```

```

/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [0
6:52:28] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

```

```

warnings.warn(smsg, UserWarning)
Predicted probabilities for the test set:
[[5.1860439e-07 5.7017002e-07 3.9342234e-07 ... 8.7231638e-07
 7.8322529e-07 5.8524296e-07]
 [8.1559443e-07 1.7004106e-06 6.5709958e-07 ... 1.0578590e-06
 1.6773043e-05 2.0757868e-06]
 [5.4707510e-07 7.7939086e-07 5.7252436e-07 ... 1.0011939e-06
 4.1819862e-06 9.0937459e-07]
 ...
 [7.2769092e-07 3.8095675e-06 7.3217876e-07 ... 6.5145679e-07
 5.3072574e-07 4.8228475e-07]
 [2.6380076e-06 2.8987354e-06 2.4175074e-06 ... 4.4621897e-06
 1.0261622e-05 3.1510957e-05]
 [8.4509293e-07 7.6321942e-07 6.7003754e-07 ... 7.0236740e-07
 3.7610098e-06 9.6112626e-07]]
Log loss for the test set: 0.00400363072165128

```

AUTOENCODER

```

In [ ]: def binary_mask(p_m, data):
        """Generates a binary mask with probability p_m for corruption."""
        return pd.DataFrame(np.random.binomial(1, p_m, data.shape), columns=data

def x_corruption(mask, data):
    """Applies corruption to the data using the mask."""
    shuffled = data.apply(lambda col: np.random.permutation(col))
    return data * (1 - mask) + shuffled * mask

```

```

In [ ]: from keras.layers import Input, Dense
        from keras.models import Model
        from keras import models
        import numpy as np

```

```

def self_supervised(x_unlabeled_scaled,p_m, alpha, parameters):

    # extract the batch_size and epochs
    epochs = parameters['epochs']
    batch_size = parameters['batch_size']
    _,dimension = x_unlabeled_scaled.shape

    # model creation
    # defining an encoder
    # auto encoder ---> corrupted input ---> encoder ----> latent space ---> decoder
    # working on the encoder part and extracting the latent space
    # creating a fully connecting network with the number of neurons in the fully connected
    # input_layer will be of size 37
    input_layer = Input(shape=(dimension,))

    #encoder model
    h = Dense(int(dimension),activation='relu')(input_layer)

    #output1 ---> mask estimation
    output1 = Dense(int(dimension) , activation='sigmoid', name='mask_estimation')(h)

    #output2 ---> feature estimation
    output2 = Dense(int(dimension) , activation='sigmoid', name='feature_estimation')(h)

    model = Model(inputs = input_layer, outputs=[output1,output2])
    model.compile(optimizer="rmsprop",loss={'mask_estimation': 'binary_crossentropy', 'feature_estimation': 'binary_crossentropy'})

    # Generate corrupted data and mask
    corruption_mask = binary_mask(p_m,x_unlabeled_scaled)
    x_unlabeled_corrupted = x_corruption(corruption_mask, x_unlabeled_scaled)
    m_label = (x_unlabeled_scaled != x_unlabeled_corrupted).astype(int) # Calculating the mask

    # Fit the model
    model.fit(x_unlabeled_corrupted,{'mask_estimation':m_label,'feature_estimation':m_label},epochs=epochs,batch_size=batch_size)

    name_of_layer = model.layers[1].name # Assuming the encoder layer is the first layer
    layer_output = model.get_layer(name_of_layer).output
    encoder = models.Model(inputs=model.input , outputs=layer_output)
    model.summary()
    return encoder

```

```

In [ ]: x_unlab = x_unlabeled_scaled


p_m=0.3


alpha= 2.0


parameters={'batch_size':128,
            'epochs':50,
            }


encoder_model =self_supervised(x_unlab,p_m, alpha, parameters)


```


Epoch 1/50
1262/1262  **4s** 2ms/step - feature_estimation_loss: 0.6400
- loss: 2.3753 - mask_estimation_loss: 1.7354


Epoch 2/50
1262/1262  **3s** 2ms/step - feature_estimation_loss: 0.6114
- loss: 2.0003 - mask_estimation_loss: 1.3889


Epoch 3/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6096
- loss: 1.9839 - mask_estimation_loss: 1.3744


Epoch 4/50
1262/1262  **3s** 2ms/step - feature_estimation_loss: 0.6089
- loss: 1.9754 - mask_estimation_loss: 1.3665


Epoch 5/50
1262/1262  **4s** 2ms/step - feature_estimation_loss: 0.6086
- loss: 1.9660 - mask_estimation_loss: 1.3574


Epoch 6/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6084
- loss: 1.9680 - mask_estimation_loss: 1.3596


Epoch 7/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6077
- loss: 1.9605 - mask_estimation_loss: 1.3527


Epoch 8/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6078
- loss: 1.9638 - mask_estimation_loss: 1.3560


Epoch 9/50
1262/1262  **4s** 3ms/step - feature_estimation_loss: 0.6075
- loss: 1.9646 - mask_estimation_loss: 1.3571


Epoch 10/50
1262/1262  **4s** 2ms/step - feature_estimation_loss: 0.6071
- loss: 1.9657 - mask_estimation_loss: 1.3586


Epoch 11/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6071
- loss: 1.9580 - mask_estimation_loss: 1.3509


Epoch 12/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6066
- loss: 1.9587 - mask_estimation_loss: 1.3521


Epoch 13/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6068
- loss: 1.9581 - mask_estimation_loss: 1.3513


Epoch 14/50
1262/1262  **4s** 3ms/step - feature_estimation_loss: 0.6063
- loss: 1.9622 - mask_estimation_loss: 1.3559


















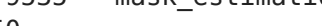
Epoch 15/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6059
- loss: 1.9578 - mask_estimation_loss: 1.3518

Epoch 16/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6059
- loss: 1.9602 - mask_estimation_loss: 1.3544

Epoch 17/50
1262/1262  **2s** 2ms/step - feature_estimation_loss: 0.6057
- loss: 1.9575 - mask_estimation_loss: 1.3519

Epoch 18/50
1262/1262  **3s** 2ms/step - feature_estimation_loss: 0.6054
- loss: 1.9564 - mask_estimation_loss: 1.3510

Epoch 19/50
1262/1262  **3s** 2ms/step - feature_estimation_loss: 0.6052

- loss: 1.9581 - mask_estimation_loss: 1.3530
Epoch 20/50
1262/1262  5s 2ms/step - feature_estimation_loss: 0.6054
- loss: 1.9546 - mask_estimation_loss: 1.3492
Epoch 21/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6053
- loss: 1.9524 - mask_estimation_loss: 1.3471
Epoch 22/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6051
- loss: 1.9545 - mask_estimation_loss: 1.3494
Epoch 23/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6050
- loss: 1.9472 - mask_estimation_loss: 1.3422
Epoch 24/50
1262/1262  4s 3ms/step - feature_estimation_loss: 0.6049
- loss: 1.9530 - mask_estimation_loss: 1.3480
Epoch 25/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6051
- loss: 1.9588 - mask_estimation_loss: 1.3537
Epoch 26/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6049
- loss: 1.9526 - mask_estimation_loss: 1.3477
Epoch 27/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6048
- loss: 1.9547 - mask_estimation_loss: 1.3499
Epoch 28/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6050
- loss: 1.9528 - mask_estimation_loss: 1.3478
Epoch 29/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6043
- loss: 1.9521 - mask_estimation_loss: 1.3479
Epoch 30/50
1262/1262  4s 2ms/step - feature_estimation_loss: 0.6047
- loss: 1.9541 - mask_estimation_loss: 1.3495
Epoch 31/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6045
- loss: 1.9515 - mask_estimation_loss: 1.3470
Epoch 32/50
1262/1262  2s 2ms/step - feature_estimation_loss: 0.6045
- loss: 1.9507 - mask_estimation_loss: 1.3462
Epoch 33/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6043
- loss: 1.9500 - mask_estimation_loss: 1.3457
Epoch 34/50
1262/1262  4s 3ms/step - feature_estimation_loss: 0.6042
- loss: 1.9546 - mask_estimation_loss: 1.3504
Epoch 35/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6044
- loss: 1.9507 - mask_estimation_loss: 1.3463
Epoch 36/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6044
- loss: 1.9535 - mask_estimation_loss: 1.3491
Epoch 37/50
1262/1262  3s 2ms/step - feature_estimation_loss: 0.6047
- loss: 1.9545 - mask_estimation_loss: 1.3497
Epoch 38/50

1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6044
 - loss: 1.9541 - mask_estimation_loss: 1.3497
 Epoch 39/50
1262/1262 ————— **4s** 3ms/step - feature_estimation_loss: 0.6043
 - loss: 1.9508 - mask_estimation_loss: 1.3465
 Epoch 40/50
1262/1262 ————— **4s** 2ms/step - feature_estimation_loss: 0.6042
 - loss: 1.9552 - mask_estimation_loss: 1.3510
 Epoch 41/50
1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6043
 - loss: 1.9457 - mask_estimation_loss: 1.3415
 Epoch 42/50
1262/1262 ————— **2s** 2ms/step - feature_estimation_loss: 0.6040
 - loss: 1.9546 - mask_estimation_loss: 1.3506
 Epoch 43/50
1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6041
 - loss: 1.9500 - mask_estimation_loss: 1.3459
 Epoch 44/50
1262/1262 ————— **4s** 3ms/step - feature_estimation_loss: 0.6043
 - loss: 1.9537 - mask_estimation_loss: 1.3494
 Epoch 45/50
1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6041
 - loss: 1.9540 - mask_estimation_loss: 1.3499
 Epoch 46/50
1262/1262 ————— **2s** 2ms/step - feature_estimation_loss: 0.6040
 - loss: 1.9502 - mask_estimation_loss: 1.3462
 Epoch 47/50
1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6039
 - loss: 1.9477 - mask_estimation_loss: 1.3438
 Epoch 48/50
1262/1262 ————— **3s** 2ms/step - feature_estimation_loss: 0.6039
 - loss: 1.9512 - mask_estimation_loss: 1.3473
 Epoch 49/50
1262/1262 ————— **4s** 3ms/step - feature_estimation_loss: 0.6038
 - loss: 1.9483 - mask_estimation_loss: 1.3445
 Epoch 50/50
1262/1262 ————— **5s** 3ms/step - feature_estimation_loss: 0.6038
 - loss: 1.9481 - mask_estimation_loss: 1.3442

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None , 36)	0
dense (Dense)	(None , 36)	1,332
mask_estimation (Dense)	(None , 36)	1,332
feature_estimation (Dense)	(None , 36)	1,332

Total params: 7,994 (31.23 KB)

Trainable params: 3,996 (15.61 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 3,998 (15.62 KB)

Saving Autoencoder

```
In [ ]: import os

# Define the path where you want to save the model
encoder_path = "content/encoder_model.keras"

# Create the directory if it doesn't exist
os.makedirs(os.path.dirname(encoder_path), exist_ok=True)

# Save the model
encoder_model.save(encoder_path)

print(f"Model saved to {encoder_path}")
```

Model saved to content/encoder_model.keras

```
In [ ]: from keras.models import load_model
encoder = load_model(encoder_path)
```

Evaluating Encoded Features with Logistic Regression and XGBoost

```
In [ ]: import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
import xgboost as xgb

# Adjust y_train and y_test labels to start from 0 by subtracting the minimum
y_train -= y_train.min()
y_test -= y_test.min()

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train) # Scale training data
x_test_scaled = scaler.transform(x_test)

# Step 1: Define the encoder model and train it on x_unlab (assumed to be done)
# For demonstration, use the encoder to transform train and test data

# Use the encoder to get the encoded data for training and testing
x_train_scaled_encoded = encoder.predict(x_train_scaled)
x_test_scaled_encoded = encoder.predict(x_test_scaled)

# Check shapes
print("Encoded x_train shape:", x_train_scaled_encoded.shape)
print("Encoded x_test shape:", x_test_scaled_encoded.shape)

# Step 2: Logistic Regression
log_reg = LogisticRegression(max_iter=1000) # Set max_iter to a higher value
log_reg.fit(x_train_scaled_encoded, y_train)
```

```

# Predict on the test set using Logistic Regression
y_encoded_log_reg = log_reg.predict_proba(x_test_scaled_encoded)

# Compute log loss for logistic regression predictions
log_reg_loss = log_loss(y_test, y_encoded_log_reg)
print("Log Loss for Logistic Regression:", log_reg_loss)

# Step 3: XGBoost Model
xgb_model = xgb.XGBClassifier(eval_metric='logloss', random_state=42)
xgb_model.fit(x_train_scaled_encoded, y_train)

# Predict on the test set using XGBoost
y_encoded_xgb = xgb_model.predict_proba(x_test_scaled_encoded)

# Compute log loss for XGBoost predictions
xgb_loss = log_loss(y_test, y_encoded_xgb)
print("Log Loss for XGBoost:", xgb_loss)

```

```

2279/2279 ————— 3s 1ms/step
977/977 ————— 2s 2ms/step
Encoded x_train shape: (72928, 36)
Encoded x_test shape: (31256, 36)
Log Loss for Logistic Regression: 0.0359260007670638
Log Loss for XGBoost: 0.06054763807339786

```

SEMI SUPERVISED

```

In [20]: import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, optimizers, losses
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import load_model
import pandas as pd

# Define the model
def model(input_dimension, hidden_dimension, label_dimension, activation=tf.nn.relu):
    inputs = tf.keras.Input(shape=input_dimension, name='model_input')
    x = layers.Dense(hidden_dimension, activation=activation, name='model_hidden1')
    x = layers.Dense(hidden_dimension, activation=activation, name='model_hidden2')
    y_logits = layers.Dense(label_dimension, activation=None, name='model_logits')
    y = layers.Activation('softmax', name='model_output')(y_logits)
    return tf.keras.Model(inputs=inputs, outputs=[y_logits, y], name="model")

# Training function
def train(feature_batch, label_batch, unlabeled_feature_batch, model, beta,
          with_tf.GradientTape() as tape:
    # Labeled data loss
    y_logits, _ = model(feature_batch, training=True)
    y_loss = supv_loss_fn(label_batch, y_logits)

    # Unlabeled data loss
    unlabeled_y_logits, _ = model(unlabeled_feature_batch, training=True)
    _, variance = tf.nn.moments(unlabeled_y_logits, axes=0)
    unlabeled_y_loss = tf.reduce_mean(variance)

```

```

        # Total loss
        total_loss = y_loss + beta * unlabeled_y_loss

    # Gradient computation and update
    grads = tape.gradient(total_loss, model.trainable_weights)
    optimizer.apply_gradients(zip(grads, model.trainable_weights))
    return total_loss

# Semi-supervised function
def semi_supervised(x_train, y_train, x_unlabeled, x_test, parameters, mask):
    # Ensure NumPy arrays
    if isinstance(x_train, pd.DataFrame):
        x_train = x_train.values
    if isinstance(y_train, pd.Series):
        y_train = y_train.values
    if isinstance(x_unlabeled, pd.DataFrame):
        x_unlabeled = x_unlabeled.values
    if isinstance(x_test, pd.DataFrame):
        x_test = x_test.values

    # Hyperparameters
    hidden_dimension = parameters['hidden_dim']
    batch_size = parameters['batch_size']
    epochs = parameters['iterations']
    input_dimension = x_train.shape[1]

    # Label preprocessing: One-hot encoding for CategoricalCrossentropy
    unique_classes = np.unique(y_train)
    label_dimension = len(unique_classes)
    class_mapping = {label: idx for idx, label in enumerate(unique_classes)}
    y_train_mapped = np.vectorize(class_mapping.get)(y_train)
    y_train_one_hot = to_categorical(y_train_mapped, num_classes=label_dimension)

    # Data splitting
    index = np.random.permutation(x_train.shape[0])
    train_index = index[:int(len(index) * 0.9)]
    valid_index = index[int(len(index) * 0.9):]
    splitted_train_x, splitted_train_y = x_train[train_index], y_train_one_hot[train_index]
    splitted_valid_x, splitted_valid_y = x_train[valid_index], y_train_one_hot[valid_index]

    # Load pre-trained encoder
    encoder = load_model(encoder_path)
    x_valid_encoded = encoder.predict(splitted_valid_x)
    x_test_encoded = encoder.predict(x_test)

    # Initialize the supervised model
    supervised_model = model(input_dimension=(encoder.output_shape[1]),
                             hidden_dimension=hidden_dimension,
                             label_dimension=label_dimension)

    optimizer = optimizers.Adam()
    supv_loss_fn = losses.CategoricalCrossentropy(from_logits=True)

    # Training loop
    for epoch in range(epochs):
        batch_index = np.random.choice(splitted_train_x.shape[0], batch_size)

```

```

        batch_x, batch_y = splitted_train_x[batch_index], splitted_train_y[batch_index]
        batch_x_encoded = encoder.predict(batch_x)

        batch_unlabeled_index = np.random.choice(x_unlabeled.shape[0], batch_size)
        batch_unlabeled_x = x_unlabeled[batch_unlabeled_index]

        batch_unlabeled_x_shuffled = []
        for _ in range(K):
            mask = np.random.binomial(1, mask_probability, batch_unlabeled_x.shape)
            corrupted_data = batch_unlabeled_x * (1 - mask) + np.random.permutation(batch_unlabeled_x[mask])
            corrupted_data_encoded = encoder.predict(corrupted_data)
            batch_unlabeled_x_shuffled.append(corrupted_data_encoded)
        batch_unlabeled_x_shuffled = np.concatenate(batch_unlabeled_x_shuffled)

        total_loss = train(batch_x_encoded, batch_y, batch_unlabeled_x_shuffled, encoder)

        y_valid_logit, _ = supervised_model(x_valid_encoded, training=False)
        y_valid_loss = supv_loss_fn(splitted_valid_y, y_valid_logit)

        if epoch % 100 == 0:
            print(f'Epoch: {epoch}/{epochs}, Validation Loss: {y_valid_loss}')

    y_test_logit, _ = supervised_model(x_test_encoded, training=False)
    return y_test_logit, supervised_model

# Hyperparameters
mask_probability = 0.3
K = 3
beta = 1.0
parameters = {
    'hidden_dim': 100,
    'batch_size': 128,
    'iterations': 1100
}

# Assuming x_train, y_train, x_unlabeled_scaled, x_test are defined
encoder_path = "content/encoder_model.keras" # Replace with your encoder path
y_test_model, model_instance = semi_supervised(x_train, y_train, x_unlabeled_scaled, x_test,
                                                parameters, mask_probability, K, beta)

```

228/228  0s 1ms/step

977/977  1s 1ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 4ms/step

Epoch: 0/1100, Validation Loss: 8.4450

4/4  0s 4ms/step

4/4  0s 3ms/step

4/4  0s 4ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 4ms/step

4/4  0s 4ms/step

4/4  0s 2ms/step

4/4  0s 4ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 4ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 4ms/step

4/4  0s 4ms/step

4/4  0s 5ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 3ms/step

4/4  0s 2ms/step

4/4  0s 3ms/step

4/4  0s 2ms/step

4/4  0s 2ms/step

4/4		0s	3ms/step
4/4		0s	2ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	2ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	6ms/step
4/4		0s	5ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step

















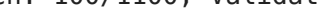










































[illegible]

[illegible]

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	9ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step

4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 4ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 4ms/step
4/4  0s 2ms/step
4/4  0s 2ms/step
Epoch: 100/1100, Validation Loss: 0.3351
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 4ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 4ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 4ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 4ms/step
4/4  0s 3ms/step
4/4  0s 3ms/step
4/4  0s 2ms/step
4/4  0s 2ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	8ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

[illegible]

4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step

4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

4/4 ————— 0s 2ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
Epoch: 200/1100, Validation Loss: 0.1962
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 8ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 4ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	7ms/step
4/4	0s	8ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	7ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step

4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
Epoch: 300/1100, Validation Loss: 0.1598			
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	5ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	8ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step

4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step

4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step

4/4	<div></div>	0s	7ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	7ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	5ms/step
Epoch: 400/1100, Validation Loss: 0.1150			
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step

4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

[illegible]

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	9ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step

4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
Epoch: 500/1100, Validation Loss: 0.1124		
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	8ms/step

4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	10ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	7ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step

4/4	0s	7ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step

4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
Epoch: 600/1100, Validation Loss: 0.0881
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 7ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 7ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step

[illegible]

[illegible]

4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
Epoch: 700/1100, Validation Loss: 0.0797			
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	5ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	6ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	2ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	4ms/step
4/4	<div></div>	0s	3ms/step
4/4	<div></div>	0s	7ms/step

4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step

4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

4/4		0s	4ms/step
4/4		0s	2ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	6ms/step
4/4		0s	4ms/step
4/4		0s	4ms/step
4/4		0s	8ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	2ms/step
4/4		0s	4ms/step
4/4		0s	5ms/step
4/4		0s	5ms/step
4/4		0s	5ms/step
4/4		0s	6ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	5ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	2ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	4ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step
4/4		0s	2ms/step
4/4		0s	3ms/step
4/4		0s	3ms/step

4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	8ms/step
4/4	0s	8ms/step
4/4	0s	10ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	8ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

[illegible]

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	8ms/step
4/4	0s	6ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
Epoch: 900/1100, Validation Loss: 0.0838		
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step

[illegible]

4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step

[illegible]

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
Epoch: 1000/1100, Validation Loss: 0.0637		
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step

4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	8ms/step
4/4	0s	6ms/step
4/4	0s	9ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step

[illegible]

4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	6ms/step
4/4	0s	8ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	7ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step

[illegible]

4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	2ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	3ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	7ms/step
4/4	0s	6ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	7ms/step
4/4	0s	8ms/step
4/4	0s	4ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	5ms/step
4/4	0s	4ms/step
4/4	0s	6ms/step
4/4	0s	6ms/step
4/4	0s	4ms/step

[illegible]

4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 6ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 7ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 7ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 3ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 2ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 5ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step
4/4 ————— 0s 4ms/step

Performace Metrics

```
In [21]: from sklearn.metrics import accuracy_score, roc_auc_score
         from sklearn.preprocessing import label_binarize
         import numpy as np

         def perf_metric(metric, y_test, y_test_hat):
             """
             Evaluate the performance of a classification model using accuracy or AUC
```

```

Parameters:
- metric (str): 'acc' for accuracy or 'auc' for AUROC.
- y_test (np.array): Ground truth labels, integer encoded, shape: (n_samples, n_classes)
- y_test_hat (np.array): Predicted probabilities, shape: (n_samples, n_classes)

Returns:
- float: Calculated performance metric.
"""
# Validate input
if metric not in ['acc', 'auc']:
    raise ValueError("Unsupported metric. Use 'acc' for accuracy or 'auc' for AUROC.")

# Accuracy metric
if metric == 'acc':
    # Convert predicted probabilities to class labels
    y_pred = np.argmax(y_test_hat, axis=1)
    return accuracy_score(y_test, y_pred)

# AUROC metric
elif metric == 'auc':
    n_classes = y_test_hat.shape[1]
    if n_classes == 2: # Binary classification
        # Use probabilities of the positive class
        y_pred_prob = y_test_hat[:, 1]
        return roc_auc_score(y_test, y_pred_prob)
    elif n_classes > 2: # Multiclass classification
        # Use one-vs-rest approach
        y_test_bin = label_binarize(y_test, classes=np.unique(y_test))
        return roc_auc_score(y_test_bin, y_test_hat, average='macro', multi_class='ovr')
    else:
        raise ValueError("AUROC is not defined for single-class tasks.")

```

```

In [22]: # Evaluate Accuracy
accuracy = perf_metric('acc', y_test, y_test_model)
print(f"Accuracy: {accuracy:.4f}")

# Evaluate AUROC
auroc = perf_metric('auc', y_test, y_test_model)
print(f"AUROC: {auroc:.4f}")

```

Accuracy: 0.9705
AUROC: 0.9955

Predictions

```

In [24]: def generate_unlabeled_predictions(x_unlab, encoder, predictor):
        """
        Generate predictions for unlabeled data using an encoder and predictor.

        Parameters:
        - x_unlab: Unlabeled feature data.
        - encoder: Pretrained encoder model to encode features.
        - predictor: Trained classification model.

```

```

Returns:
- y_unlab_pred: Predicted labels for unlabeled data.
"""
# Encode unlabeled data
x_unlab_encoded = encoder.predict(x_unlab)

# Predict with the classifier
_, y_unlab_hat = predictor(x_unlab_encoded, training=False)

# Convert probabilities to predicted class labels
y_unlab_pred = np.argmax(y_unlab_hat, axis=1)
return y_unlab_pred

# Generate predictions for the unlabeled data
y_unlab_pred = generate_unlabeled_predictions(x_unlabeled_scaled, encoder, m
print(f"Predicted Labels for Unlabeled Data:\n{y_unlab_pred}")

```

5046/5046 ————— 20s 4ms/step

Predicted Labels for Unlabeled Data:

[9 9 9 ... 7 9 9]

TSNE after semi-supervised

In [23]: `pip install openTSNE`

Collecting openTSNE

Downloading openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.8 kB)

Requirement already satisfied: numpy>=1.16.6 in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.26.4)

Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.5.2)

Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from openTSNE) (1.13.1)

Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->openTSNE) (1.4.2)

Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->openTSNE) (3.5.0)

Downloading openTSNE-1.0.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)

3.0/3.0 MB 22.1 MB/s eta 0:00:00

Installing collected packages: openTSNE

Successfully installed openTSNE-1.0.2

In [26]: `from matplotlib.colors import ListedColormap`
`import matplotlib.pyplot as plt`
`from openTSNE import TSNE`
`import numpy as np`
`from matplotlib import cm`

`def plot_tsne_custom_color(features, labels, title="t-SNE Visualization"):`
 `"""`
 `Generate t-SNE visualization using OpenTSNE with a custom color scheme.`
 `Parameters:`

```

- features: The feature matrix (e.g., encoded or raw features).
- labels: Labels corresponding to the features (should be integers or ca
- title: Title of the plot.
"""

# Perform t-SNE with OpenTSNE
tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
tsne_result = tsne.fit(features)

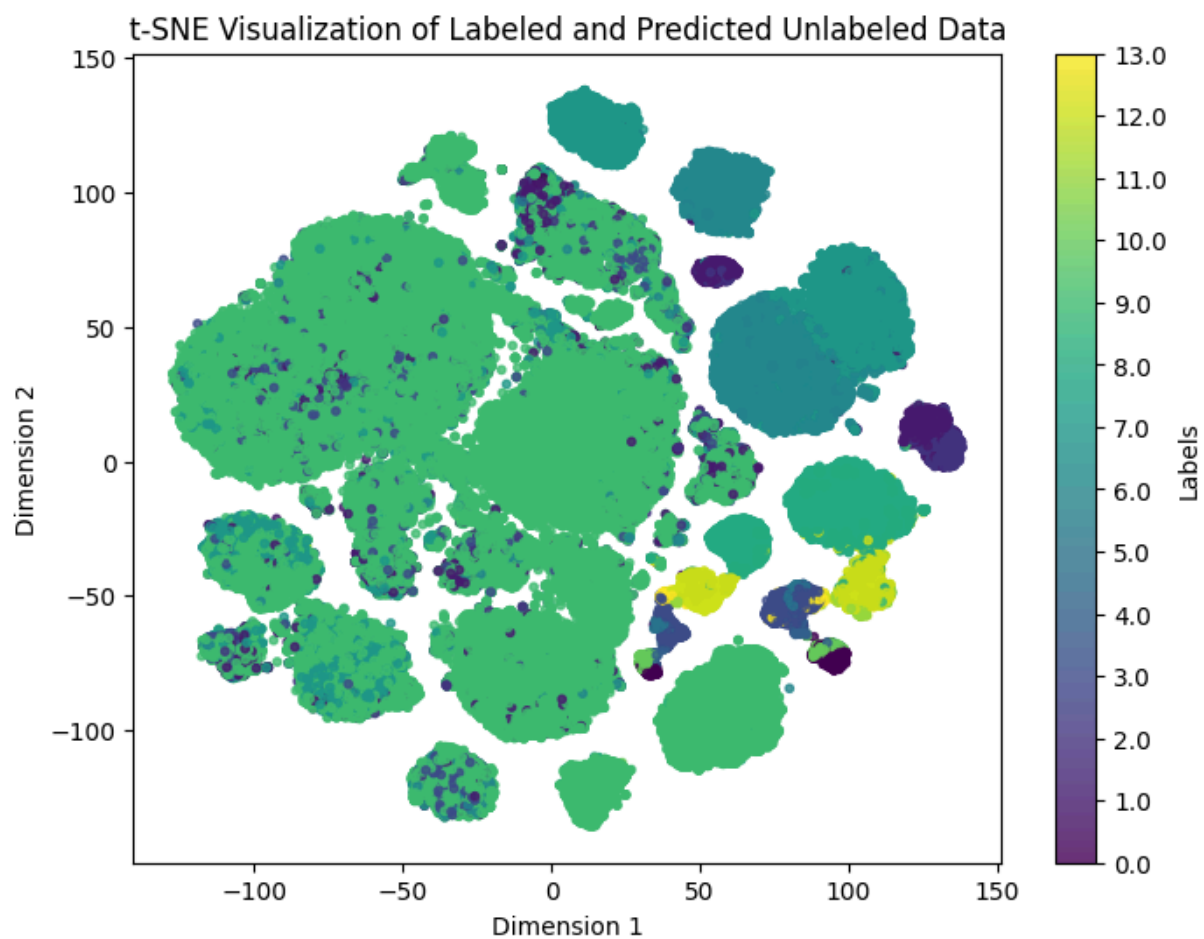
# Use a continuous colormap such as 'viridis' to match the color scheme
cmap = plt.cm.viridis

# Plot the results
plt.figure(figsize=(8, 6))
scatter = plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=labels, cm
plt.title(title)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
cbar = plt.colorbar(scatter, ticks=np.unique(labels))
cbar.set_label('Labels')
cbar.ax.set_yticklabels([str(label) for label in np.unique(labels)])
plt.show()

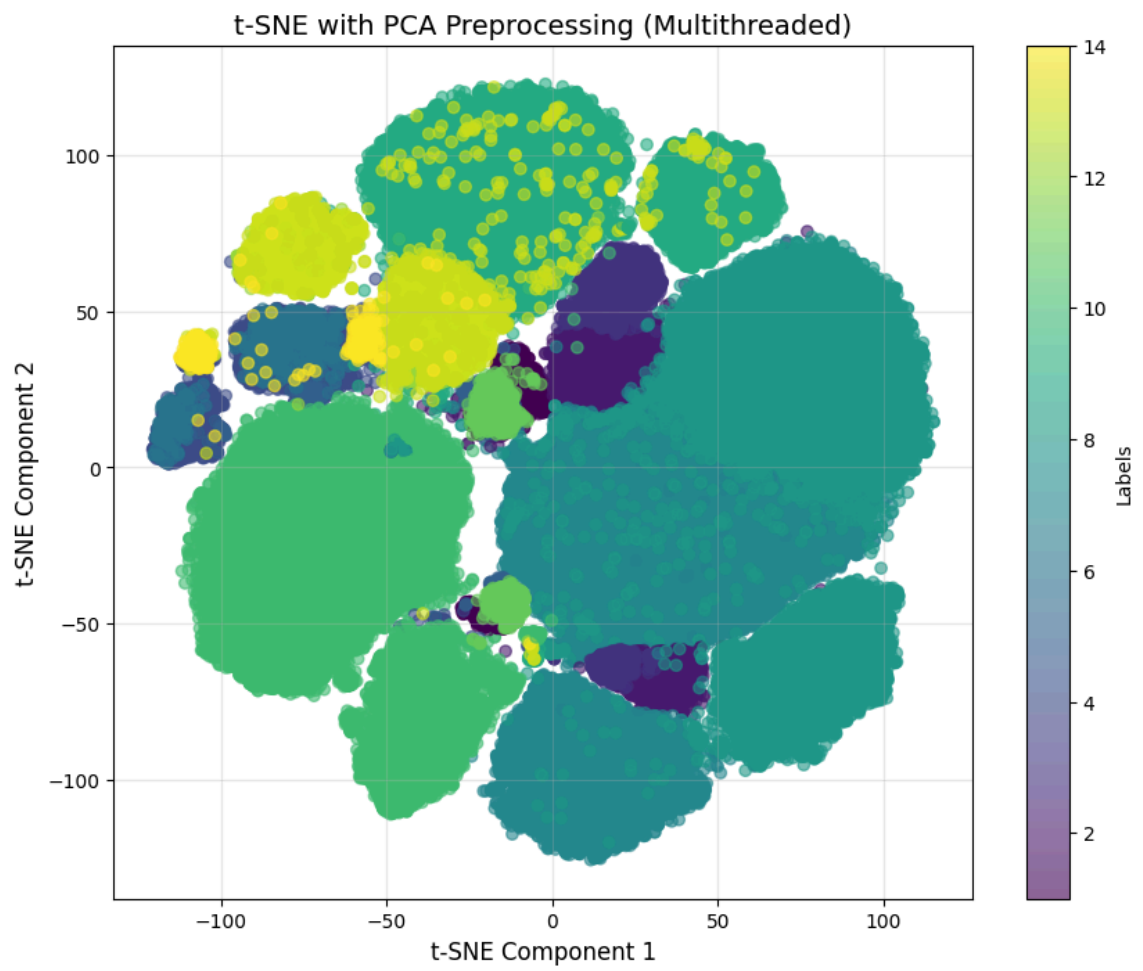
# Features (scaled unlabeled data) and predicted labels
scaled_features = np.vstack([x_train_scaled_encoded, x_unlabeled_scaled]) #
combined_labels = np.hstack([y_train, y_unlab_pred]) # Combine true and pre

# Call the t-SNE plotting function
plot_tsne_custom_color(scaled_features, combined_labels, title="t-SNE Visual

```



Intial TSNE for Comparison



```
In [28]: !pip install gradio
```

Collecting gradio
 Downloading gradio-5.6.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
 Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
 Downloading fastapi-0.115.5-py3-none-any.whl.metadata (27 kB)
Collecting ffmpeg (from gradio)
 Downloading ffmpeg-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting gradio-client==1.4.3 (from gradio)
 Downloading gradio_client-1.4.3-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.27.2)
Requirement already satisfied: huggingface-hub>=0.25.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.26.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.4)
Collecting markupsafe~=2.0 (from gradio)
 Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.10.11)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (11.0.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.9.2)
Collecting pydub (from gradio)
 Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart==0.0.12 (from gradio)
 Downloading python_multipart-0.0.12-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.2.2 (from gradio)
 Downloading ruff-0.8.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<1.0,>=0.1.1 (from gradio)
 Downloading safehttpx-0.1.1-py3-none-any.whl.metadata (4.1 kB)
Collecting semantic-version~=2.0 (from gradio)
 Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
 Downloading starlette-0.41.3-py3-none-any.whl.metadata (6.0 kB)
Collecting tomlkit==0.12.0 (from gradio)
 Downloading tomlkit-0.12.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.13.0)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.12.2)
Collecting uvicorn>=0.14.0 (from gradio)
 Downloading uvicorn-0.32.1-py3-none-any.whl.metadata (6.6 kB)

Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==1.4.3->gradio) (2024.10.0)
Collecting websockets<13.0,>=10.0 (from gradio-client==1.4.3->gradio)
 Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.14.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (3.16.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (4.66.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (2.23.4)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.18.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.25.1->gradio) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.25.1->gradio) (2.2.3)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio) (0.1.2)
Downloading gradio-5.6.0-py3-none-any.whl (57.1 MB)
57.1/57.1 MB 12.5 MB/s eta 0:00:

```

00
Downloading gradio_client-1.4.3-py3-none-any.whl (320 kB)
_____ 320.1/320.1 kB 17.6 MB/s eta 0:0
0:00
Downloading python_multipart-0.0.12-py3-none-any.whl (23 kB)
Downloading tomkit-0.12.0-py3-none-any.whl (37 kB)
Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.5-py3-none-any.whl (94 kB)
_____ 94.9/94.9 kB 6.1 MB/s eta 0:00:0
0
Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Downloading ruff-0.8.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.1 MB)
_____ 11.1/11.1 MB 97.2 MB/s eta 0:00:
00
Downloading safehttpx-0.1.1-py3-none-any.whl (8.4 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.41.3-py3-none-any.whl (73 kB)
_____ 73.2/73.2 kB 3.8 MB/s eta 0:00:0
0
Downloading uvicorn-0.32.1-py3-none-any.whl (63 kB)
_____ 63.8/63.8 kB 4.1 MB/s eta 0:00:0
0
Downloading ffmpeg-0.4.0-py3-none-any.whl (5.8 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (130 kB)
_____ 130.2/130.2 kB 8.3 MB/s eta 0:0
0:00
Installing collected packages: pydub, websockets, uvicorn, tomkit, semantic-version, ruff, python-multipart, markupsafe, ffmpeg, aiofiles, starlette, safehttpx, gradio-client, fastapi, gradio
  Attempting uninstall: markupsafe
    Found existing installation: MarkupSafe 3.0.2
    Uninstalling MarkupSafe-3.0.2:
      Successfully uninstalled MarkupSafe-3.0.2
Successfully installed aiofiles-23.2.1 fastapi-0.115.5 ffmpeg-0.4.0 gradio-5.6.0 gradio-client-1.4.3 markupsafe-2.1.5 pydub-0.25.1 python-multipart-0.0.12 ruff-0.8.0 safehttpx-0.1.1 semantic-version-2.10.0 starlette-0.41.3 tomkit-0.12.0 uvicorn-0.32.1 websockets-12.0

```

```

In [ ]: import gradio as gr
import pandas as pd
import numpy as np
from openTSNE import TSNE
import matplotlib.pyplot as plt
from matplotlib import colormaps
from tensorflow.keras.models import load_model

# Function to generate predictions for unlabeled data
def generate_predictions_for_unlabeled(x_unlab, encoder, predictor):
    """Encode data and generate predictions for unlabeled samples."""
    encoded_data = encoder.predict(x_unlab) # Encode data using the encoder
    predictions = predictor(encoded_data, training=False) # Get the predicted
    predicted_classes = np.argmax(predictions, axis=1) # Get predicted clas

```

```

    return predicted_classes

# Function for t-SNE visualization
def create_tsne_visualization(features, labels, title="t-SNE Visualization")
    """Perform t-SNE dimensionality reduction and visualize with distinct cl
    tsne = TSNE(n_components=2, perplexity=30, n_iter=1000, random_state=42)
    tsne_result = tsne.fit(features)

    unique_labels = np.unique(labels)
    label_to_index = {label: idx for idx, label in enumerate(unique_labels)}
    color_indices = np.array([label_to_index[label] for label in labels])

    cmap = colormaps.get_cmap('tab10') # Get the colormap
    fig, ax = plt.subplots(figsize=(8, 6))
    scatter = ax.scatter(
        tsne_result[:, 0],
        tsne_result[:, 1],
        c=color_indices,
        cmap=cmap,
        s=5,
        alpha=0.7
    )
    ax.set_title(title)
    ax.set_xlabel('t-SNE Dimension 1')
    ax.set_ylabel('t-SNE Dimension 2')

    # Add legend for cluster labels
    legend_handles = [
        plt.Line2D([], [], marker='o', color=cmap(idx / len(unique_labels)),
        for idx in range(len(unique_labels))
    ]
    ax.legend(legend_handles, unique_labels, title="Clusters", loc="best", b

    return fig

# Function to process and visualize predictions and t-SNE
def process_and_visualize_data(start_row, end_row):
    """
    Process a subset of the x_unlabeled dataset, predict labels, and generat
    """

    global x_unlabeled # Ensure x_unlabeled is loaded

    # Parse input row indices
    start_row = int(start_row)
    end_row = int(end_row)

    # Select subset of data
    x_subset = x_unlabeled[start_row:end_row]

    # Load pre-trained encoder and predictor
    encoder = load_model(encoder_path)
    trained_model = encoder
    predictor = trained_model # Assume predictor is preloaded

    # Predict labels for the subset

```

```

predicted_labels = generate_predictions_for_unlabeled(x_subset, encoder,

# Create t-SNE visualization
tsne_figure = create_tsne_visualization(x_subset, predicted_labels, titl

# Return visualization and predictions
return tsne_figure, pd.DataFrame({"Predicted Labels": predicted_labels})

# Set up Gradio interface
inputs = [
    gr.Number(label="Start Row", value=3, precision=0), # Starting row input
    gr.Number(label="End Row", value=109, precision=0) # Ending row input
]

outputs = [
    gr.Plot(label="t-SNE Visualization"), # t-SNE plot
    gr.Dataframe(label="Predicted Labels (Top 10)") # Dataframe for predictions
]

gr.Interface(
    fn=process_and_visualize_data,
    inputs=inputs,
    outputs=outputs,
    title="Self-Supervised Learning Visualization",
    description="Generate predictions and visualize data with t-SNE."
).launch(debug=True)

```

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().

* Running on public URL: <https://36750f0ce1196f02fb.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (<https://huggingface.co/spaces>)

Self-Supervised Learning Visualization

Generate predictions and visualize data with t-SNE.

Start Row

3

End Row

109

Clear

Submit

 t-SNE Visualization



4/4  0s 26ms/step

This notebook was converted with convert.ploomber.io